Cat. No. W451-E1-02

SYSMAC CP Series
CP1H-X40D□-□
CP1H-XA40D□-□
CP1H-Y20DT-D

CP1H CPU Unit

PROGRAMMING MANUAL

OMRON

CP1H-X40D□-□
CP1H-XA40D□-□
CP1H-Y20DT-D

CP1H CPU Unit

Programming Manual

Revised May 2006

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 “PLC” means Programmable Controller. “PC” is used, however, in some CX-Programmer displays to mean Programmable Controller.

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.

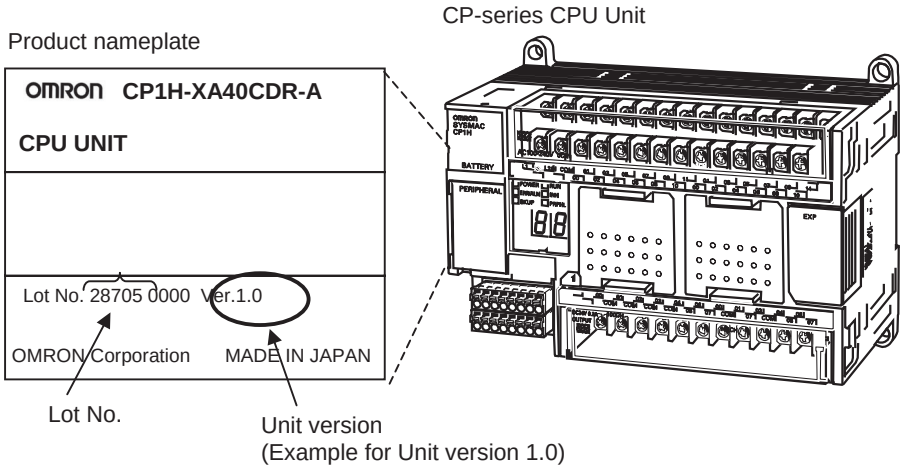
Unit Versions of CP-series CPU Units

Unit Versions

A “unit version” has been introduced to manage CPU Units in the CP Series according to differences in functionality accompanying Unit upgrades.

Notation of Unit Versions on Products

The unit version is given to the right of the lot number on the nameplate of the products for which unit versions are being managed, as shown below.



Confirming Unit Versions with Support Software

CX-Programmer version 6.1 or higher can be used to confirm the unit version using one of the following two methods. (See note.)

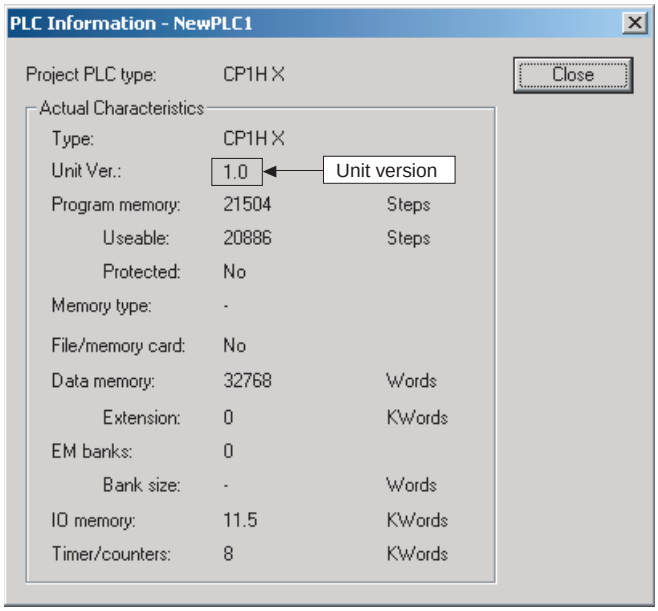
- Using the ***PLC Information***
- Using the ***Unit Manufacturing Information***

Note CX-Programmer version 6.1 or lower cannot be used to confirm unit versions for CP-series CPU Units.

PLC Information

- If you know the device type and CPU type, select them in the *Change PLC* Dialog Box, go online, and select ***PLC - Edit - Information*** from the menus.
- If you don't know the device type and CPU type but are connected directly to the CPU Unit on a serial line, select ***PLC - Auto Online*** to go online, and then select ***PLC - Edit - Information*** from the menus.

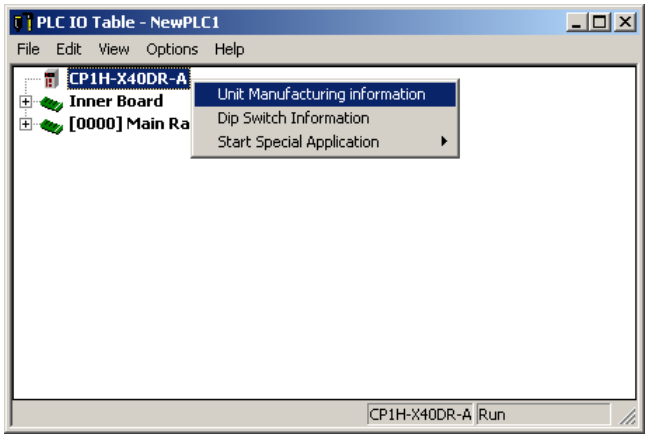
In either case, the following *PLC Information* Dialog Box will be displayed.



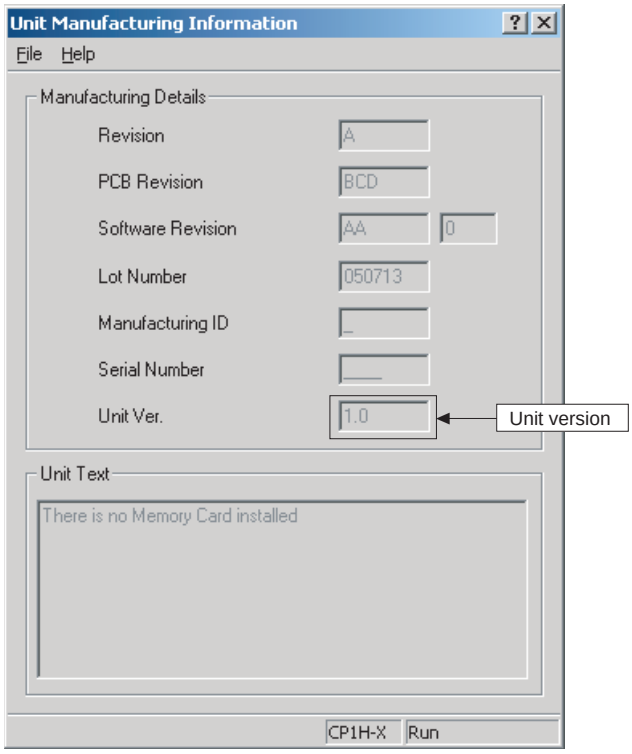
Use the above display to confirm the unit version of the CPU Unit.

Unit Manufacturing Information

In the IO Table Window, right-click and select ***Unit Manufacturing information - CPU Unit.***



The following *Unit Manufacturing information* Dialog Box will be displayed.



Use the above display to confirm the unit version of the CPU Unit connected online.

Using the Unit Version Labels

The following unit version labels are provided with the CPU Unit.

These labels can be attached to the front of previous CPU Units to differentiate between CPU Units of different unit versions.

TABLE OF CONTENTS

TABLE OF CONTENTS

About this Manual:

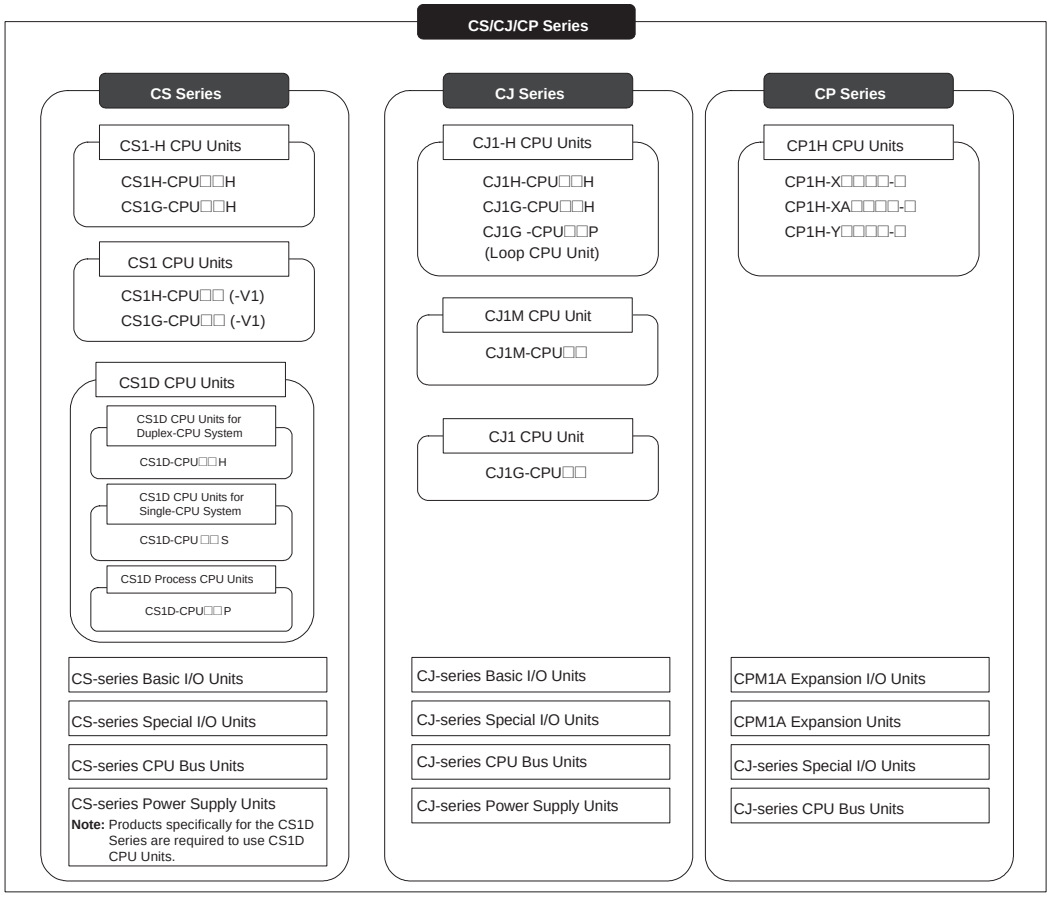
This manual describes programming the CP-series Programmable Controllers (PLCs) and includes the sections described below. The CP Series provides advanced package-type PLCs based on OMRON's advanced control technologies and vast experience in automated control.

Please read this manual carefully and be sure you understand the information provided before attempting to install or operate a CP-series PLC. Be sure to read the precautions provided in the following section.

Definition of the CP Series

The CP Series is centered around the CP1H CPU Units and is designed with the same basic architecture as the CS and CJ Series. The Special I/O Units and CPU Bus Units of the CJ Series can thus be used. CJ-series Basic I/O Units, however, cannot be used. Always use CPM1A Expansion Units and CPM1A Expansion I/O Units when expanding I/O capacity.

I/O words are allocated in the same way as the CPM1A/CPM2A PLCs, i.e., using fixed areas for inputs and outputs.



Precautions provides general precautions for using the Programmable Controller and related devices.

Section 1 describes the basic concepts required to program the CP1H.

Section 2 describes the operation of tasks and how to use tasks in programming.

Section 3 describes each of the instructions that can be used in programming CP-series PLCs. Instructions are described in order of function.

Section 4 lists the execution times and number of steps for all instructions supported by the CP1H PLCs, and describes the execution times for function block instances.

The ***Appendices*** provide lists of the programming instructions in order of function and in order of function number.

Related Manuals

The following manuals are used for the CP-series CPU Units. Refer to these manuals as required.

Cat. No.	Model numbers	Manual name	Description
W450	CP1H-X40D□-□ CP1H-XA40D□-□ CP1H-Y20DT-D	SYSMAC CP Series CP1H CPU Unit Operation Manual	Provides the following information on the CP Series: • Overview, design, installation, maintenance, and other basic specifications • Features • System configuration • Mounting and wiring • I/O memory allocation • Troubleshooting Use this manual together with the *CP1H Programmable Controllers Programming Manual* (W451).
W451	CP1H-X40D□-□ CP1H-XA40D□-□ CP1H-Y20DT-D	SYSMAC CP Series CP1H CPU Unit Programming Manual	Provides the following information on the CP Series: • Programming instructions • Programming methods • Tasks • File memory • Functions Use this manual together with the *CP1H Programmable Controllers Operation Manual* (W450).
W342	CS1G/H-CPU□□H CS1G/H-CPU□□-V1 CS1D-CPU□□H CS1D-CPU□□S CS1W-SCU21 CS1W-SCB21-V1/41-V1 CJ1G/H-CPU□□H CJ1G-CPU□□P CP1H-CPU□□ CJ1G-CPU□□ CJ1W-SCU21-V1/41-V1	SYSMAC CS/CJ-series Communications Commands Reference Manual	Describes commands addressed to CS-series, and CJ-series CPU Units, including C-mode commands and FINS commands. **Note** This manual describes on commands address to CPU Units regardless of the communications path. (CPU Unit serial ports, Serial Communications Unit/Board ports, and Communications Unit ports can be used.) Refer to the relevant operation manuals for information on commands addresses to Special I/O Units and CPU Bus Units.
W446	WS02-CXPC1-E-V70	SYSMAC CX-Programmer Ver. 7.0 Operation Manual	Provides information on installing and operating the CX-Programmer for all functions except for function blocks.
W447	WS02-CXPC1-E-V70	SYSMAC CX-Programmer Ver. 7.0 Operation Manual Function Blocks	Provides specifications and operating procedures for function blocks. Function blocks can be used with CX-Programmer Ver. 6.1 or higher and either a CS1-H/CJ1-H CPU Unit with a unit version of 3.0 or a CP1H CPU Unit. Refer to W446 for operating procedures for functions other than function blocks.
W444	CXONE-AL□□C-E	CX-One FA Integrated Tool Package Setup Manual	Provides an overview of the CX-One FA Integrated Tool and installation procedures.
W445	CXONE-AL□□C-E	CX-Integrator Operation Manual	Describes CX-Integrator operating procedures and provides information on network configuration (data links, routing tables, Communications Units setup, etc.
W344	WS02-PSTC1-E	CX-Protocol Operation Manual	Provides operating procedures for creating protocol macros (i.e., communications sequences) with the CX-Protocol and other information on protocol macros. The CX-Protocol is required to create protocol macros for user-specific serial communications or to customize the standard system protocols.

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 CP-series Programmable Controllers (PLCs) and related devices.

The information contained in this section is important for the safe and reliable application of Programmable Controllers. You must read this section and understand the information contained before attempting to set up or operate a PLC system.

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, 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.

This manual provides information for programming and operating the Unit. Be sure to read this manual before attempting to use the Unit and keep this manual close at hand for reference during operation.

⚠WARNING It is extremely important that a PLC and all PLC Units 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 a PLC System to the above-mentioned applications.

3 Safety Precautions

⚠WARNING Do not attempt to take any Unit 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 Units. 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 Programmable Controller), including the following items, to ensure safety in the system if an abnormality occurs due to malfunction of the PLC or another external factor affecting the PLC 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 PLC 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 PLC or outputs may remain ON or OFF due to deposits on or burning of the output relays, or 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-V DC output (service power supply to the PLC) 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.

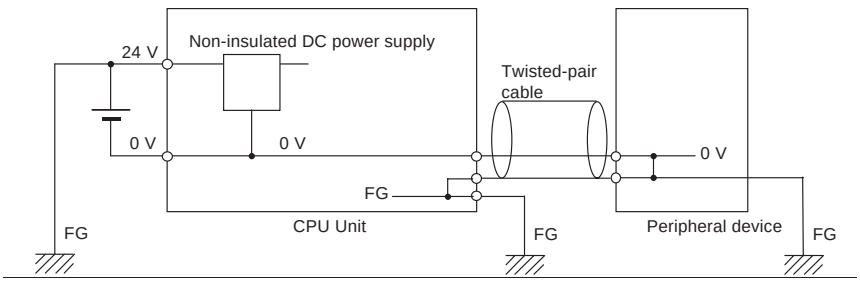
⚠Caution Confirm safety at the destination node before transferring a program to another node or editing the I/O area. Doing either of these without confirming safety may result in injury.

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

⚠Caution Do not touch anywhere near the power supply parts or I/O terminals while the power is ON, and immediately after turning OFF the power. The hot surface may cause burn injury.

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

⚠Caution When connecting the PLC to a computer or other peripheral device, either ground the 0 V side of the external power supply or do not ground the external power supply at all. Otherwise the external power supply may be shorted depending on the connection methods of the peripheral device. DO NOT ground the 24 V side of the external power supply, as shown in the following diagram.



⚠**Caution** After programming (or reprogramming) using the IOWR instruction, confirm that correct operation is possible with the new ladder program and data before starting actual operation. Any irregularities may cause the product to stop operating, resulting in unexpected operation in machinery or equipment.

⚠**Caution** The CP1H CPU Units automatically back up the user program and parameter data to flash memory when these are written to the CPU Unit. I/O memory (including the DM Area, Counter present values and Completion Flags, and HR Area), however, is not written to flash memory. The DM Area, Counter present values and Completion Flags, and HR Area can be held during power interruptions with a battery. If there is a battery error, the contents of these areas may not be accurate after a power interruption. If the contents of the DM Area, Counter present values and Completion Flags, and HR Area are used to control external outputs, prevent inappropriate outputs from being made whenever the Battery Error Flag (A402.04) is ON.

4 Operating Environment Precautions

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

- 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 PLC 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 PLC 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.

5 Application Precautions

Observe the following precautions when using the PLC System.

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

- Always connect to 100 Ω or less when installing the Units. Not connecting to a ground of 100 Ω or less may result in electric shock.
- Always turn OFF the power supply to the PLC before attempting any of the following. Not turning OFF the power supply may result in malfunction or electric shock.
  - Mounting or dismounting Expansion Units or any other Units
  - Connecting or removing the Memory Cassette or Option Board
  - Setting DIP switches or rotary 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 PLC or the system, or could damage the PLC or PLC Units. Always heed these precautions.

- Install external breakers and take other safety measures against short-circuiting in external wiring. Insufficient safety measures against short-circuiting may result in burning.
- Mount the Unit only after checking the connectors and terminal blocks completely.
- 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.
- Wire all connections correctly according to instructions in this manual.
- 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 label attached to the Unit when wiring. Removing the label may result in malfunction.
- Remove the 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 input terminals in excess of the rated input voltage. Excess voltages may result in burning.
- Do not apply voltages or connect loads to the output terminals in excess of the maximum switching capacity. Excess voltage or loads may result in burning.

- Be sure that the terminal blocks, connectors, Option Boards, and other items with locking devices are properly locked into place. Improper locking may result in malfunction.
- 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 Unit. Not checking the program may result in an unexpected operation.
- Resume operation only after transferring to the new CPU Unit and/or Special I/O Units the contents of the DM, HR, and CNT Areas required for resuming operation. Not doing so may result in an unexpected operation.
- 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 PLC (including the setting of the startup operating mode).
  - Force-setting/force-resetting any bit in memory.
  - Changing the present value of any word or any set value in memory.
- 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.
- 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.
- Before touching the Unit, be sure to first touch a grounded metallic object in order to discharge any static buildup. Not doing so may result in malfunction or damage.
- Do not touch the Expansion I/O Unit Connecting Cable while the power is being supplied in order to prevent malfunction due to static electricity.
- Do not turn OFF the power supply to the Unit while data is being transferred.
- 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.
- Double-check the pin numbers when assembling and wiring the connectors.
- Wire correctly according to specified procedures.
- Do not connect pin 6 (+5V) on the RS-232C Option Board on the CPU Unit to any external device other than the NT-AL001 or CJ1W-CIF11 Conversion Adapter. The external device and the CPU Unit may be damaged.
- Use the dedicated connecting cables specified in this manual to connect the Units. Using commercially available RS-232C computer cables may cause failures in external devices or the CPU Unit.

- Check that data link tables and parameters are properly set before starting operation. Not doing so may result in unexpected operation. Even if the tables and parameters are properly set, confirm that no adverse effects will occur in the system before running or stopping data links.
- Transfer a routing table to the CPU Unit only after confirming that no adverse effects will be caused by restarting CPU Bus Units, which is automatically done to make the new tables effective.
- The user program and parameter area data in the CPU Unit is backed up in the built-in flash memory. The BKUP indicator will light on the front of the CPU Unit when the backup operation is in progress. Do not turn OFF the power supply to the CPU Unit when the BKUP indicator is lit. The data will not be backed up if power is turned OFF.
- Do not turn OFF the power supply to the PLC while the Memory Cassette is being accessed. Doing so may corrupt the data in the Memory Cassette. The 7-segment LED will light to indicate writing progress while the Memory Cassette is being accessed. Wait for the LED display to go out before turning OFF the power supply to the PLC.
- Before replacing the battery, supply power to the CPU Unit for at least 5 minutes and then complete battery replacement within 5 minutes of turn OFF the power supply. Memory data may be corrupted if this precaution is not observed.
- Always use the following size wire when connecting I/O Units, Special I/O Units, and CPU Bus Units: AWG22 to AWG18 (0.32 to 0.82 mm^2).
- UL standards required that batteries be replaced only by experienced technicians. Do not allow unqualified persons to replace batteries. Also, always follow the replacement procedure provided in the manual.
- Never short-circuit the positive and negative terminals of a battery or charge, disassemble, heat, or incinerate the battery. Do not subject the battery to strong shocks or deform the barry by applying pressure. Doing any of these may result in leakage, rupture, heat generation, or ignition of the battery. Dispose of any battery that has been dropped on the floor or otherwise subjected to excessive shock. Batteries that have been subjected to shock may leak if they are used.
- Always construct external circuits so that the power to the PLC it turned ON before the power to the control system is turned ON. If the PLC power supply is turned ON after the control power supply, temporary errors may result in control system signals because the output terminals on DC Output Units and other Units will momentarily turn ON when power is turned ON to the PLC.
- Fail-safe measures must be taken by the customer to ensure safety in the event that outputs from Output Units remain ON as a result of internal circuit failures, which can occur in relays, transistors, and other elements.
- If the I/O Hold Bit is turned ON, the outputs from the PLC will not be turned OFF and will maintain their previous status when the PLC is switched from RUN or MONITOR mode to PROGRAM mode. Make sure that the external loads will not produce dangerous conditions when this occurs. (When operation stops for a fatal error, including those produced with the FALS(007) instruction, all outputs from Output Unit will be turned OFF and only the internal output status will be maintained.)

- Dispose of the product and batteries according to local ordinances as they apply.
  Have qualified specialists properly dispose of used batteries as industrial waste.



「廢電池請回收」

6 Conformance to EC Directives

6-1 Applicable Directives

- EMC Directives
- Low Voltage Directive

6-2 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 The applicable EMC (Electromagnetic Compatibility) standard is EN61131-2.

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 PLC (EN61131-2).

6-3 Conformance to EC Directives

The CP1H PLCs comply with EC Directives. To ensure that the machine or device in which the CP1H PLC is used complies with EC Directives, the PLC must be installed as follows:

1,2,3...

1. The CP1H PLC must be installed within a control panel.
2. You must use reinforced insulation or double insulation for the DC power supplies used for I/O Units and CPU Units requiring DC power. The output holding time must be 10 ms minimum for the DC power supply connected to the power supply terminals on Units requiring DC power.
3. CP1H PLCs complying with EC Directives also conform to EN61131-2. 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.

6-4 Relay Output Noise Reduction Methods

The CP1H PLCs conforms to the Common Emission Standards (EN61131-2) 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 PLC.

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

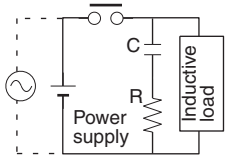
Countermeasures are not required if the frequency of load switching for the whole system with the PLC included is less than 5 times per minute.

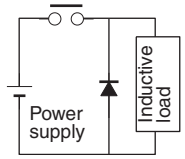
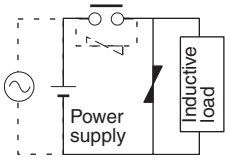
Countermeasures are required if the frequency of load switching for the whole system with the PLC included is more than 5 times per minute.

Note Refer to EN61131-2 for more details.

Countermeasure Examples

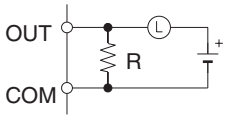
When switching an inductive load, connect an surge protector, diodes, etc., in parallel with the load or contact as shown below.

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.	---

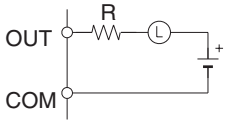
When switching a load with a high inrush current such as an incandescent lamp, suppress the inrush current as shown below.

Countermeasure 1



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

Countermeasure 2



Providing a limiting resistor

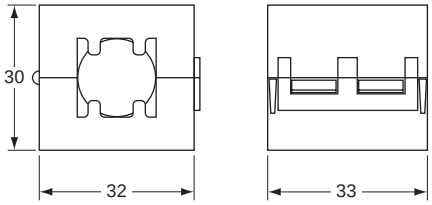
6-5 Conditions for Meeting EMC Directives when Using CPM1A Relay Expansion I/O Units

EN61131-2 immunity testing conditions when using the CPM1A-40EDR or CPM1A-16ER with an CP1W-CN811 I/O Connecting Cable are given below.

Recommended Ferrite Core

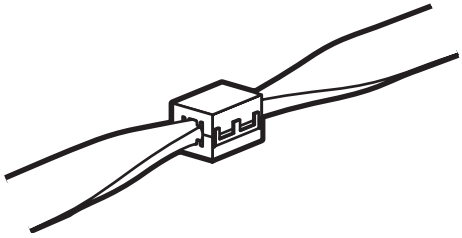
Ferrite Core (Data Line Filter): 0443-164151 manufactured by Nisshin Electric

Minimum impedance: 90 Ω at 25 MHz, 160 Ω at 100 MHz

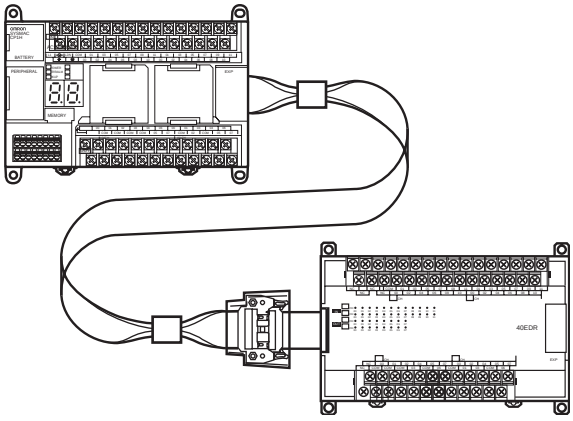


Recommended Connection Method

1,2,3... 1. Cable Connection Method



2. Connection Method
   As shown below, connect a ferrite core to each end of the CP1W-CN811 I/O Connecting Cable.



SECTION 1
Programming Concepts

This section describes the basic concepts required to program the CP1H.

1-1 Programming Concepts

1-1-1 Programs and Tasks

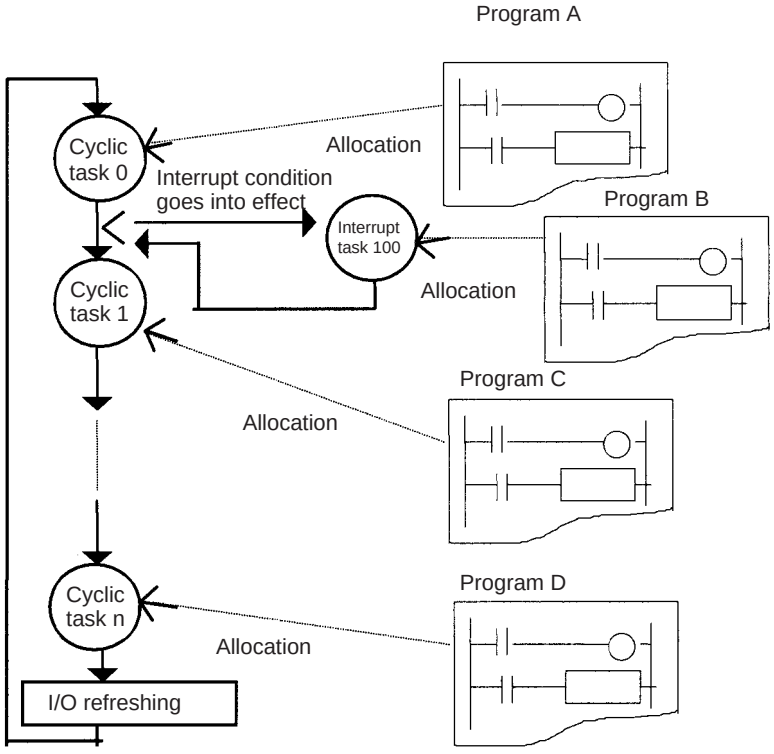
Tasks specify the sequence and interrupt conditions under which individual programs will be executed. They are broadly grouped into the following types:

1,2,3... 1. Tasks executed sequentially that are called cyclic tasks.

2. Tasks executed by interrupt conditions that are called interrupt tasks.

Note Interrupt tasks can be executed cyclically in the same way as cyclic tasks. These are called “extra cyclic tasks.”

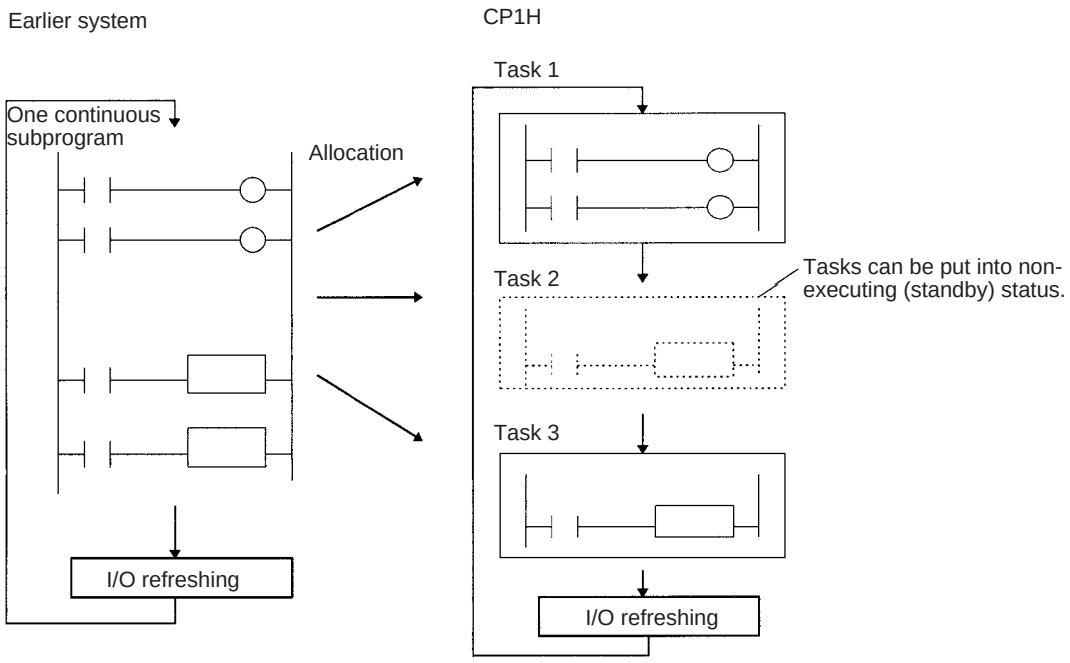
Programs allocated to cyclic tasks will be executed sequentially by task number and I/O will be refreshed once per cycle after all tasks (more precisely tasks that are in executable status) are executed. If an interrupt condition goes into effect during processing of the cyclic tasks, the cyclic task will be interrupted and the program allocated to the interrupt task will be executed.



In the above example, programming would be executed in the following order: start of A, B, remainder of A, C, and then D. This assumes that the interrupt condition for interrupt task 100 was established during execution of program A. When execution of program B is completed, the rest of program A would be executed from the place where execution was interrupted.

With earlier OMRON PLCs, one continuous program is formed from several continuous parts. The programs allocated to each task are single programs that terminate with an END instruction, just like the single program in earlier PLCs.

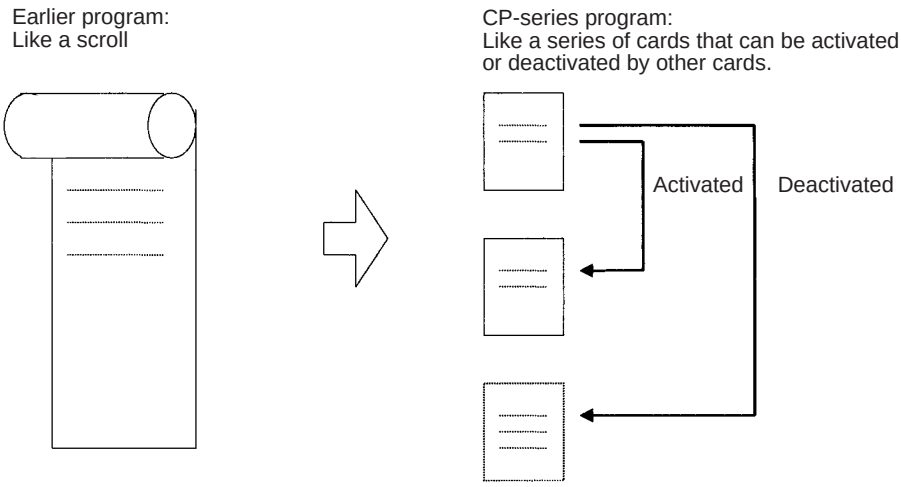
One feature of the cyclic tasks is that they can be enabled (executable status) and disabled (standby status) by the task control instructions. This means that several program components can be assembled as a task, and that only specific programs (tasks) can then be executed as needed for the current product model or process being performed (program step switching). Therefore performance (cycle time) is greatly improved because only required programs will be executed as needed.



A task that has been executed will be executed in subsequent cycles, and a task that is on standby will remain on standby in subsequent cycles unless it is executed again from another task.

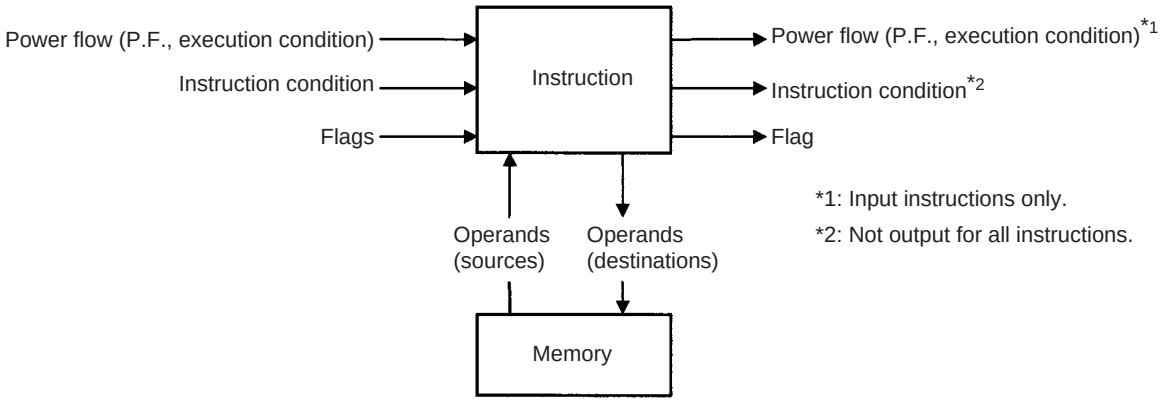
Note Unlike earlier programs that can be compared to reading a scroll, tasks can be compared to reading through a series of individual cards.

- All cards are read in a preset sequence starting from the lowest number.
- All cards are designated as either active or inactive, and cards that are inactive will be skipped. (Cards are activated or deactivated by task control instructions.)
- A card that is activated will remain activated and will be read in subsequent sequences. A card that is deactivated will remain deactivated and will be skipped until it is reactivated by another card.



1-1-2 Basic Information on Instructions

Programs consist of instructions. The conceptual structure of the inputs to and outputs from an instruction is shown in the following diagram.

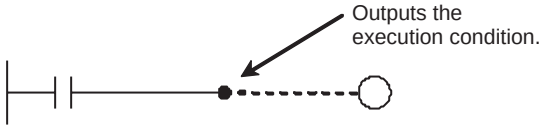


Power Flow

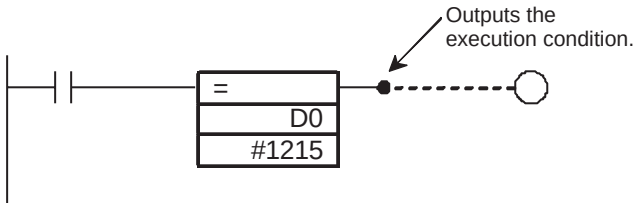
The power flow is the execution condition that is used to control the execute and instructions when programs are executing normally. In a ladder program, power flow represents the status of the execution condition.

Input Instructions

- Load instructions indicate a logical start and outputs the execution condition.

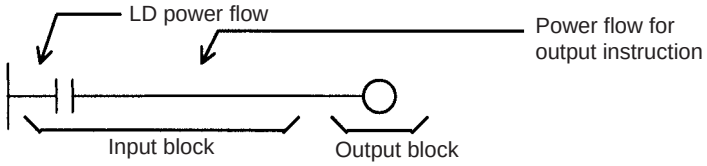


- Intermediate instructions input the power flow as an execution condition and output the power flow to an intermediate or output instruction.



Output Instructions

Output instructions execute all functions, using the power flow as an execution condition.



Instruction Conditions

Instruction conditions are special conditions related to overall instruction execution that are output by the following instructions. Instruction conditions have a higher priority than power flow (P.F.) when it comes to deciding whether or not to execute an instruction. An instruction may become not be executed or may act differently depending on instruction conditions. Instruction conditions are reset (canceled) at the start of each task, i.e., they are reset when the task changes.

The following instructions are used in pairs to set and cancel certain instruction conditions. These paired instructions must be in the same task.

Instruction condition	Description	Setting instruction	Canceling instruction
Interlocked	An interlock turns OFF part of the program. Special conditions, such as turning OFF output bits, resetting timers, and holding counters are in effect.	IL(002)	ILC(003)
BREAK(514) execution	Ends a FOR(512) - NEXT(513) loop during execution. (Prevents execution of all instructions until to the NEXT(513) instruction.)	BREAK(514)	NEXT(513)
	Executes a JMP0(515) to JME0(516) jump.	JMP0(515)	JME0(516)
Block program execution	Executes a program block from BPRG(096) to BEND(801).	BPRG(096)	BEND(801)

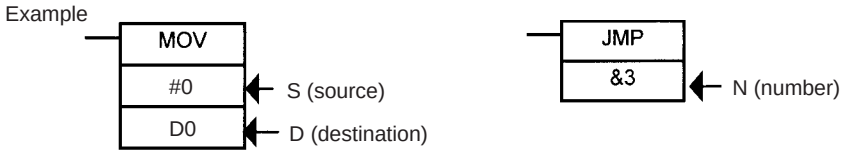
Flags

In this context, a flag is a bit that serves as an interface between instructions.

Input flags	Output flags
• Differentiation Flags Differentiation result flags. The status of these flags are input automatically to the instruction for all differentiated up/down output instructions and the DIFU(013)/DIFD(014) instructions. • Carry (CY) Flag The Carry Flag is used as an unspecified operand in data shift instructions and addition/subtraction instructions. • Flags for Special Instructions These include teaching flags for FPD(269) instructions and network communications enabled flags	• Differentiation Flags Differentiation result flags. The status of these flags are output automatically from the instruction for all differentiated up/down output instructions and the UP(521)/DOWN(522) instruction. • Condition Flags Condition Flags include the Always ON/OFF Flags, as well as flags that are updated by results of instruction execution. In user programs, these flags can be specified by labels, such as ER, CY, >, =, A1, A0, rather than by addresses. • Flags for Special Instructions These include MSG(046) execution completed flags.

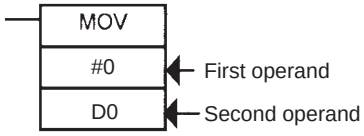
Operands

Operands specify preset instruction parameters (boxes in ladder diagrams) that are used to specify I/O memory area contents or constants. An instruction can be executed entering an address or constant as the operands. Operands are classified as source, destination, or number operands.



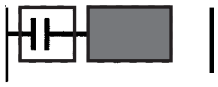
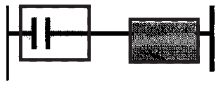
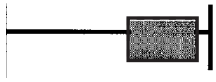
Operand types		Operand symbol	Description	
Source	Specifies the address of the data to be read or a constant.	S	Source Operand	Source operand other than control data (C)
		C	Control data	Compound data in a source operand that has different meanings depending bit status.
Destination (Results)	Specifies the address where data will be written.	D (R)	---	
Number	Specifies a particular number used in the instruction, such as a jump number or subroutine number.	N	---	

Note Operands are also called the first operand, second operand, and so on, starting from the top of the instruction.



1-1-3 Instruction Location and Execution Conditions

The following table shows the possible locations for instructions. Instructions are grouped into those that do and those do not require execution conditions.

Instruction type		Possible location	Execution condition	Diagram	Examples
Input instructions	Logical start (Load instructions)	Connected directly to the left bus bar or is at the beginning of an instruction block.	Not required.		LD, LD TST(350), LD > (and other symbol comparison instructions)
	Intermediate instructions	Between a logical start and the output instruction.	Required.		AND, OR, AND TEST(350), AND > (and other ADD symbol comparison instructions), UP(521), DOWN(522), NOT(520), etc.
Output instructions		Connected directly to the right bus bar.	Required.		Most instructions including OUT and MOV(021).
			Not required.		END(001), JME(005), FOR(512), ILC(003), etc.

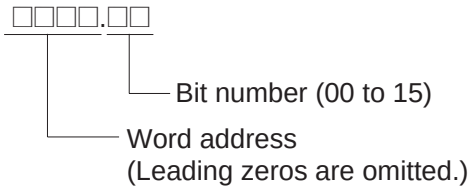
Note

(1) There is another group of instruction that executes a series of mnemonic instructions based on a single input. These are called block programming instructions. Refer to the *CP-series CP1H CPU Unit Programming Manual* for details on these block programs.

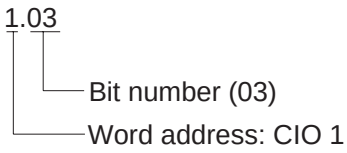
(2) If an instruction requiring an execution condition is connected directly to the left bus bar without a logical start instruction, a program error will occur when checking the program on a CX-Programmer.

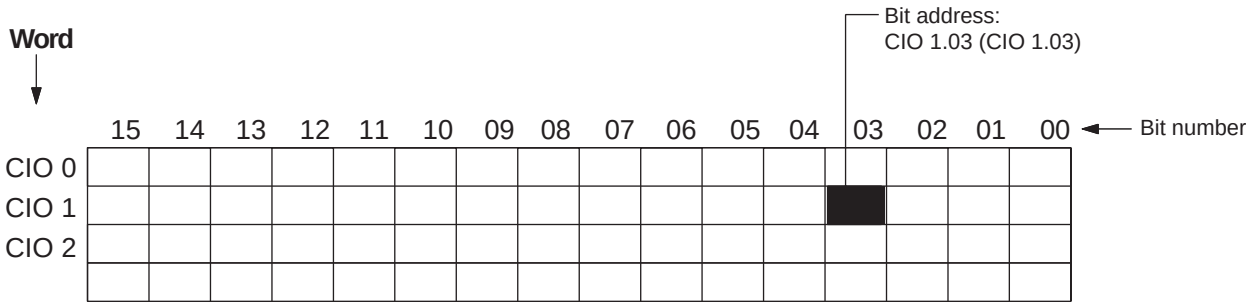
1-1-4 Addressing I/O Memory Areas

Bit Addresses

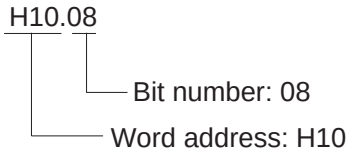


Example: The address of bit 03 in word 0001 in the CIO Area would be as shown below. This address is given as “CIO 1.03” in this manual.

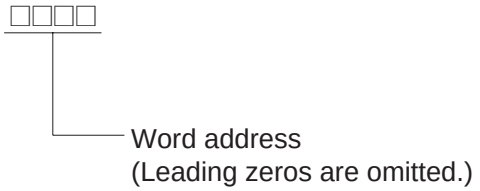




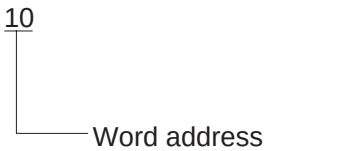
Example: Bit 08 in word H010 in the HR Area is given as shown below.



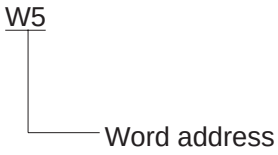
Word Addresses



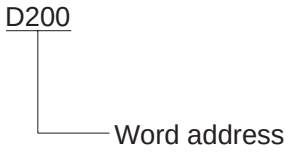
Example: The address of word 0010 (bits 00 to 15) in the CIO Area is given as shown below. This address is given as “CIO 10” in this manual.



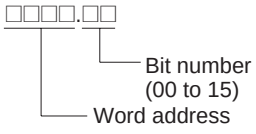
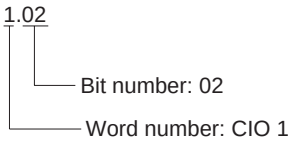
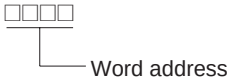
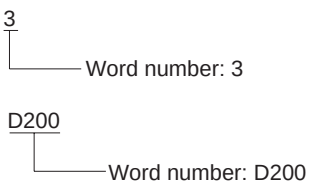
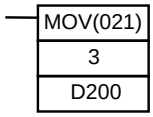
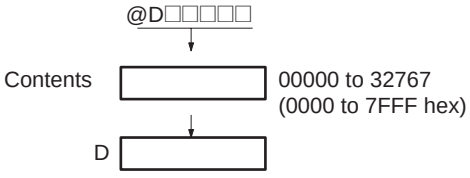
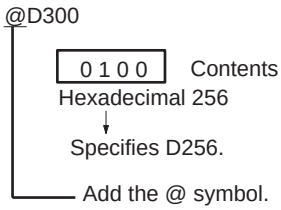
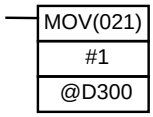
Example: The address of word W5 (bits 00 to 15) in the Work Area is given as shown below.

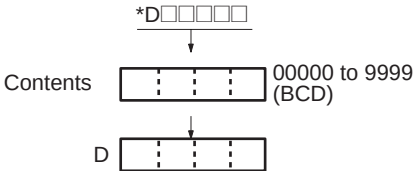


Example: The address of word D200 (bits 00 to 15) in the DM Area is given as shown below.

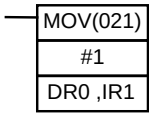
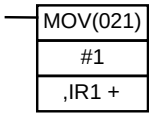
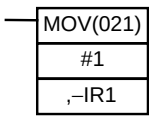


1-1-5 Specifying Instruction Operands

Operand	Description	Notation	Application examples
Specifying bit addresses	The word and bit numbers are specified directly to specify a bit (input bits).  **Note** The same addresses are used to access timer/counter Completion Flags and Present Values. There is also only one address for a Task Flag.		
Specifying word addresses	The word number is specified directly to specify the 16-bit word. 		
Specifying indirect DM addresses in Binary Mode	The offset from the beginning of the area is specified. The contents of the address will be treated as binary data (00000 to 32767) to specify the word address in Data Memory (DM). Add the @ symbol at the front to specify an indirect ad-dress in Binary Mode. 		

Operand	Description	Notation	Application examples
Specifying indirect DM addresses in BCD Mode	The offset from the beginning of the area is specified. The contents of the address will be treated as BCD data (0000 to 9999) to specify the word address in Data Memory (DM). Add an asterisk (*) at the front to specify an indirect address in BCD Mode. 	*D200 0 1 0 0 Contents BCD: 100 Specifies D100. Add an asterisk (*).	MOV(021) #1 *D200
Specifying a register directly	An index register (IR) or a data register (DR) is specified directly by specifying IR□ (□: 0 to 15) or DR□ (□: 0 to 15).	IR0	MOVR(560) 1.02 IR0 Stores the PLC memory address for CIO 10 in IR0.
		IR1	MOVR(560) 10 IR1 Stores the PLC memory address for CIO 10 in IR1.

Operand	Description		Notation	Application examples
Specifying an indirect address using a register	Indirect address (No offset)	The bit or word with the PLC memory address contained in IR□ will be specified. Specify ,IR□ to specify bits and words for instruction operands.	,IR0	,IR0 Loads the bit with the PLC memory address in IR0.
			,IR1	MOV(021) #1 ,IR1 Stores #0001 in the word with the PLC memory in IR1.
	Constant offset	The bit or word with the PLC memory address in IR□ + or – the constant is specified. Specify +/– *constant* ,IR□. Constant offsets range from –2048 to +2047 (decimal). The offset is converted to binary data when the instruction is executed.	+5,IR0	+5,IR0 Loads the bit with the PLC memory address in IR0 + 5.
			+31,IR1	MOV(021) #1 +31 ,IR1 Stores #0001 in the word with the PLC memory address in IR1 + 31

Operand	Description		Notation	Application examples
Specifying an indirect address using a register	DR offset	The bit or word with the PLC memory address in IR□ + the contents of DR□ is specified. Specify DR□ ,IR□. DR (data register) contents are treated as signed-binary data. The contents of IR□ will be given a negative offset if the signed binary value is negative.	DR0 ,IR0 DR0 ,IR1	DR0,IR0  Loads the bit with the PLC memory address in IR0 + the value in DR0.  Stores #0001 in the word with the PLC memory address in IR1 + the value in DR0.
	Auto Increment	The contents of IR□ is incremented by +1 or +2 after referencing the value as an PLC memory address. +1: Specify ,IR□+ +2: Specify ,IR□ + +	,IR0 ++ ,IR1 +	,IR0 ++  Increments the contents of IR0 by 2 after the bit with the PLC memory address in IR0 is loaded.  Increments the contents of IR1 by 1 after #0001 is stored in the word with the PLC memory address in IR1.
	Auto Decrement	The contents of IR□ is decremented by –1 or –2 after referencing the value as an PLC memory address. –1: Specify ,–IR□ –2: Specify ,– –IR□	,– –IR0 ,–IR1	,—IR  After decrementing the contents of IR0 by 2, the bit with the PLC memory address in IR0 is loaded.  After decrementing the contents of IR1 by 1, #0001 is stored in the word with the PLC memory address in IR1.

Data	Operand	Data form	Symbol	Range	Application example
16-bit constant	All binary data or a limited range of binary data	Unsigned binary	#	#0000 to #FFFF	MOV(021) #5A D100
		Signed decimal	±	–32768 to +32767	+(400) D200 –128 D300
		Unsigned decimal	&	&0 to &65535	CMP(020) D400 &999
	All BCD data or a limited range of BCD data	BCD	#	#0000 to #9999	–B(414) D500 #2000 D600
32-bit constant	All binary data or a limited range of binary data	Unsigned binary	#	#00000000 to #FFFFFFFF	MOVL(498) #17FFF D100
		Signed binary	+	–2147483648 to +2147483647	+L(401) D200 –65536 D300
		Unsigned decimal	& (See Note.)	&0 to &429467295	CMPL(060) D400 &99999
	All BCD data or a limited range of BCD data	BCD	#	#00000000 to #99999999	–BL(415) D500 #1000000 D600

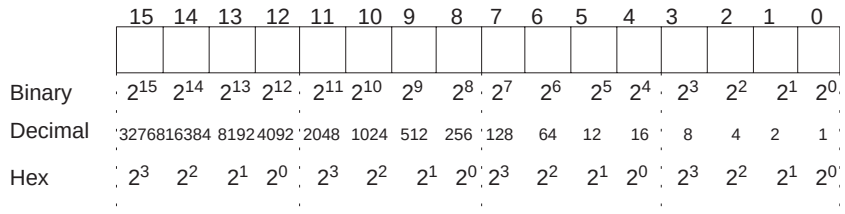
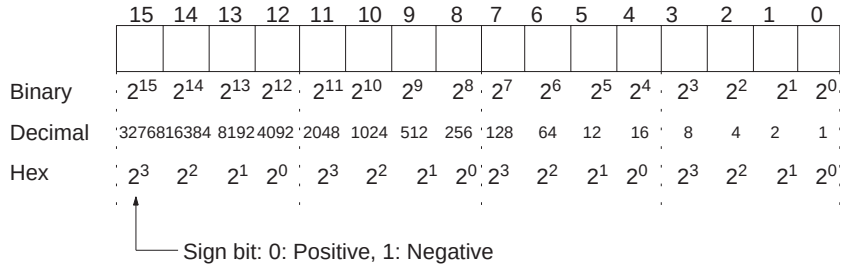
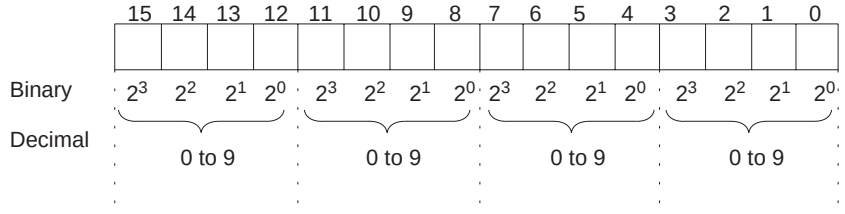
Data	Operand	Data form	Symbol	Range	Application example
Text string	**Description**		**Symbol**	**Examples**	
	Text string data is stored in ASCII (one byte except for special characters) in order from the leftmost to the rightmost byte and from the rightmost (smallest) to the leftmost word. 00 hex (NUL code) is stored in the rightmost byte of the last word if there is an odd number of characters. 0000 hex (2 NUL codes) is stored in the leftmost and rightmost vacant bytes of the last word + 1 if there is an even number of characters.		---	↓ '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	MOV$(664) D100 D200 D100 41 42 D101 43 44 D102 45 00 ↓ D200 41 42 D201 43 44 D202 45 00
	ASCII characters that can be used in a text string includes alphanumeric characters, Katakana and symbols (except for special characters). The characters are shown in the following table.				

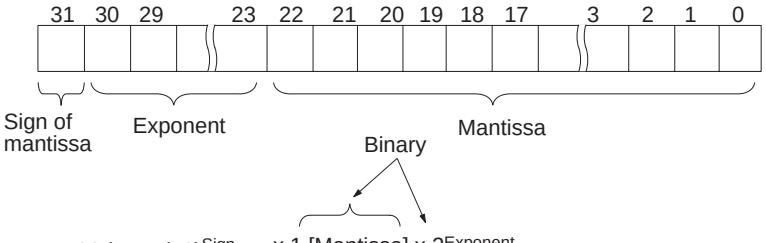
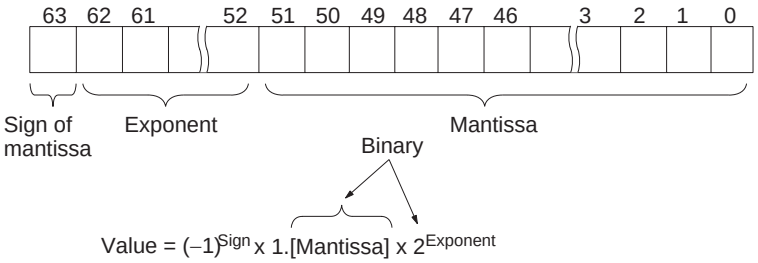
ASCII Characters

Bits 0 to 3		Bits 4 to 7															
Binary		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0			Space	0	@	P	`	p				ｰ	ﾀ	ﾐ		
0001	1			!	1	A	Q	a	q			｡	ｱ	ﾁ	ﾑ		
0010	2			"	2	B	R	b	r			｢	ｲ	ﾂ	ﾒ		
0011	3			#	3	C	S	c	s			｣	ｳ	ﾃ	ﾓ		
0100	4			$	4	D	T	d	t			､	ｴ	ﾄ	ﾔ		
0101	5			%	5	E	U	e	u			･	ｵ	ﾅ	ﾕ		
0110	6			&	6	F	V	f	v			ｦ	ｶ	ﾆ	ﾖ		
0111	7			'	7	G	W	g	w			ｧ	ｷ	ﾇ	ﾗ		
1000	8			(	8	H	X	h	x			ｨ	ｸ	ﾈ	ﾘ		
1001	9			)	9	I	Y	i	y			ｩ	ｹ	ﾉ	ﾙ		
1010	A			*	:	J	Z	j	z			ｪ	ｺ	ﾊ	ﾚ		
1011	B			+	;	K	[	k	{			ｫ	ｻ	ﾋ	ﾛ		
1100	C			,	<	L	¥	l	|			ｬ	ｼ	ﾌ	ﾜ		
1101	D			-	=	M	]	m	}			ｭ	ｽ	ﾍ	ﾝ		
1110	E			.	>	N	^	n	~			ｮ	ｾ	ﾎ	ﾞ		
1111	F			/	?	O	_	o				ｯ	ｿ	ﾏ	ﾟ		

1-1-6 Data Formats

The following table shows the data formats that the CP Series can handle.

Data type	Data format	Decimal	4-digit hexadecimal
Unsigned binary		0 to 65535	0000 to FFFF
Signed binary		0 to –32768 0 to +32767	Negative: 8000 to FFFF Positive: 0000 to 7FFF
BCD (binary coded decimal)		0 to 9999	0000 to 9999

Data type	Data format	Decimal	4-digit hexadecimal
Single-precision floating-point decimal	 Value = $(-1)^{Sign}$ x 1.[Mantissa] x $2^{Exponent}$ Sign (bit 31): 1: negative or 0: positive Mantissa: The 23 bits from bit 00 to bit 22 contain the mantissa, i.e., the portion below the decimal point in 1.□□□....., in binary. Exponent: The 8 bits from bit 23 to bit 30 contain the exponent. The exponent is expressed in binary as 127 plus n in 2^n. **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 52 bits from bit 00 to bit 51 contain the mantissa, i.e., the portion below the decimal point in 1.□□□....., in binary. Exponent: The 11 bits from bit 52 to bit 62 contain the exponent The exponent is expressed in binary as 1023 plus n in 2^n. **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 Data

In signed binary data, the leftmost bit indicates the sign of binary 16-bit data. The value is expressed in 4-digit hexadecimal.

Positive Numbers: A value is positive or 0 if the leftmost bit is 0 (OFF). In 4-digit hexadecimal, this is expressed as 0000 to 7FFF hex.

Negative Numbers: A value is negative if the leftmost bit is 1 (ON). In 4-digit hexadecimal, this is expressed as 8000 to FFFF hex. The absolute of the negative value (decimal) is expressed as a two's complement.

Example: To treat –19 in decimal as signed binary, 0013 hex (the absolute value of 19) is subtracted from FFFF hex and then 0001 hex is added to yield FFED hex.

	F	F	F	F
	1111	1111	1111	1111
True number	0	0	1	3
	0000	0000	0001	0011
—)				
	F	F	E	C
	1111	1111	1110	1100
	0	0	0	1
	0000	0000	0000	0001
+)				
Two's complement	F	F	E	D
	1111	1111	1110	1101

Complements

Generally the complement of base x refers to a number produced when all digits of a given number are subtracted from x – 1 and then 1 is added to the rightmost digit. (Example: The ten's complement of 7556 is 9999 – 7556 + 1 = 2444.) A complement is used to express a subtraction and other functions as an addition.

Example: With 8954 – 7556 = 1398, 8954 + (the ten's complement of 7556) = 8954 + 2444 = 11398. If we ignore the leftmost bit, we get a subtraction result of 1398.

Two's Complements

A two's complement is a base-two complement. Here, we subtract all digits from 1 (2 – 1 = 1) and add one.

Example: The two's complement of binary number 1101 is 1111 (F hex) – 1101 (D hex) + 1 (1 hex) = 0011 (3 hex). The following shows this value expressed in 4-digit hexadecimal.

The two's complement b hex of a hex is FFFF hex – a hex + 0001 hex = b hex. To determine the two's complement b hex of "a hex," use b hex = 10000 hex – a hex.

Example: to determine the two's complement of 3039 hex, use 10000 hex – 3039 hex = CFC7 hex.

Similarly use a hex = 10000 hex – b hex to determine the value a hex from the two's complement b hex.

Example: To determine the real value from the two's complement CFC7 hex use 10000 hex – CFC7 hex = 3039 hex.

The CP Series has two instructions: NEG(160)(2'S COMPLEMENT) and NEGL(161) (DOUBLE 2'S COMPLEMENT) that can be used to determine the two's complement from the true number or to determine the true number from the two's complement.

Signed BCD Data

Signed BCD data is a special data format that is used to express negative numbers in BCD. Although this format is found in applications, it is not strictly defined and depends on the specific application. The CP Series supports the following instructions to convert the data formats: SIGNED BCD-TO-BINARY: BINS(470), DOUBLE SIGNED BCD-TO-BINARY: BISL(472), SIGNED BINARY-TO-BCD: BCDS(471), and DOUBLE SIGNED BINARY-TO-BCD: BDSL(473). Refer to the *CP-series CPU Unit Programming Manual (W451)* for more information.

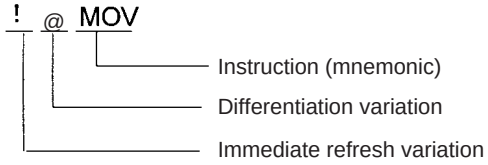
Decimal	Hexadecimal	Binary	BCD	
0	0	0000	0000	
1	1	0001	0001	
2	2	0010	0010	
3	3	0011	0011	
4	4	0100	0100	
5	5	0101	0101	
6	6	0110	0110	
7	7	0111	0111	
8	8	1000	1000	
9	9	1001	1001	
10	A	1010	0001	0000
11	B	1011	0001	0001
12	C	1100	0001	0010
13	D	1101	0001	0011
14	E	1110	0001	0100
15	F	1111	0001	0101
16	10	10000	0001	0110

Decimal	Unsigned binary (4-digit hexadecimal)	Signed binary (4-digit hexadecimal)
+65,535	FFFF	Cannot be expressed.
+65534	FFFE	
. . .	. . .	
+32,769	8001	
+32,768	8000	
+32,767	7FFF	7FFF
+32,766	7FFE	7FFE
. . .	. . .	. . .
+2	0002	0002
+1	0001	0001
0	0000	0000
–1	Cannot be expressed.	FFFF
–2		FFFE
. . .		. . .
–32,767		8001
–32,768		8000

1-1-7 Instruction Variations

The following variations are available for instructions to differentiate executing conditions and to refresh data when the instruction is executed (immediate refresh).

Variation		Symbol	Description
Differentiation	ON	@	Instruction that differentiates when the execution condition turns ON.
	OFF	%	Instruction that differentiates when the execution condition turns OFF.
Immediate refreshing		!	Refreshes data in the I/O area specified by the operands or the Special I/O Unit words when the instruction is executed.



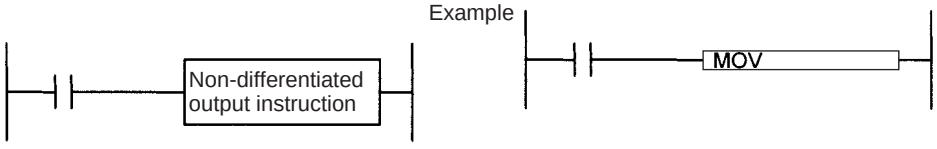
1-1-8 Execution Conditions

The CP Series offers the following types of basic and special instructions.

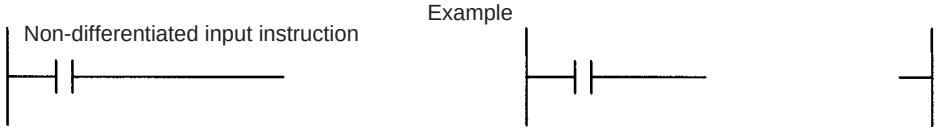
- Non-differentiated instructions executed every cycle
- Differentiated instructions executed only once

Non-differentiated Instructions

Output instructions that required execution conditions are executed once every cycle while the execution condition is valid (ON or OFF).



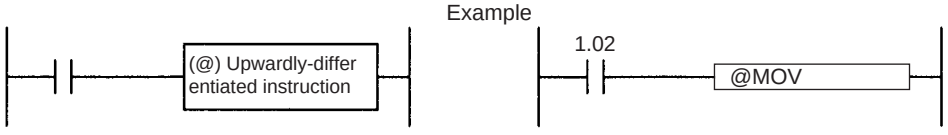
Input instructions that create logical starts and intermediate instructions read bit status, make comparisons, test bits, or perform other types of processing every cycle. If the results are ON, power flow is output (i.e., the execution condition is turned ON).



Input-differentiated Instructions

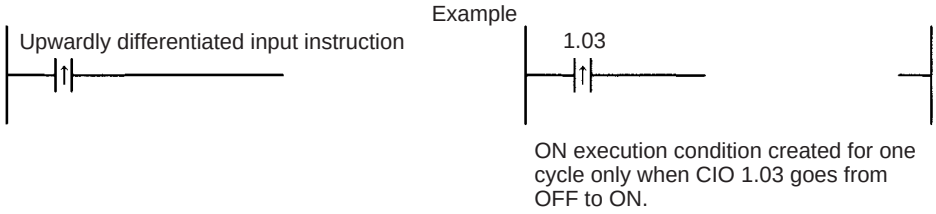
Upwardly Differentiated Instructions (Instruction Preceded by @)

- **Output Instructions**: The instruction is executed only during the cycle in which the execution condition turned ON (OFF → ON) and are not executed in the following cycles.

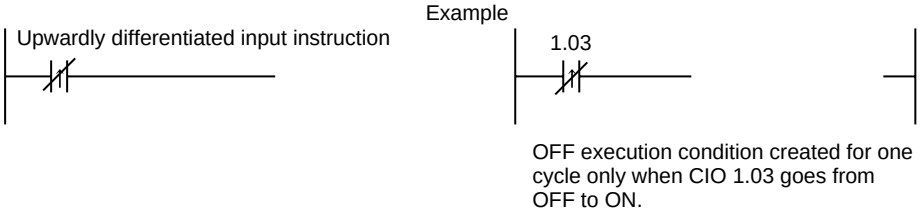


Executes the MOV instruction once when CIO 1.02 goes OFF → ON.

- **Input Instructions (Logical Starts and Intermediate Instructions)**: The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output an ON execution condition (power flow) when results switch from OFF to ON. The execution condition will turn OFF the next cycle.

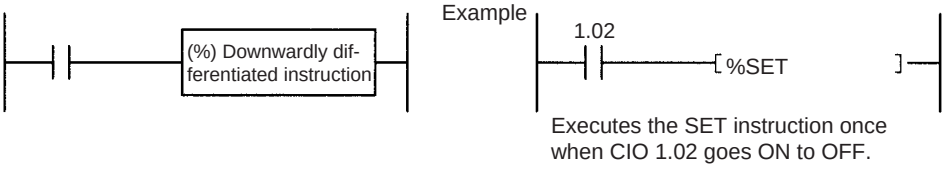


- **Input Instructions (Logical Starts and Intermediate Instructions)**: The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output an OFF execution condition (power flow stops) when results switch from OFF to ON. The execution condition will turn ON the next cycle.

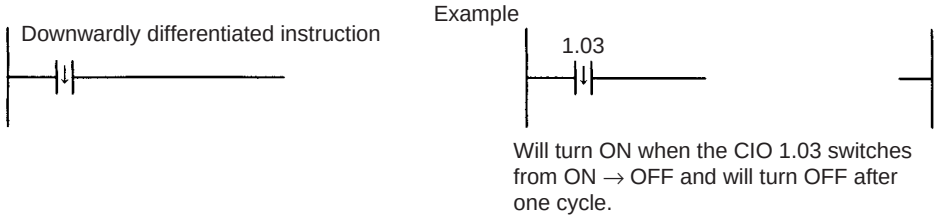


Downwardly Differentiated Instructions (Instruction preceded by %)

- **Output instructions**: The instruction is executed only during the cycle in which the execution condition turned OFF (ON → OFF) and is not executed in the following cycles.

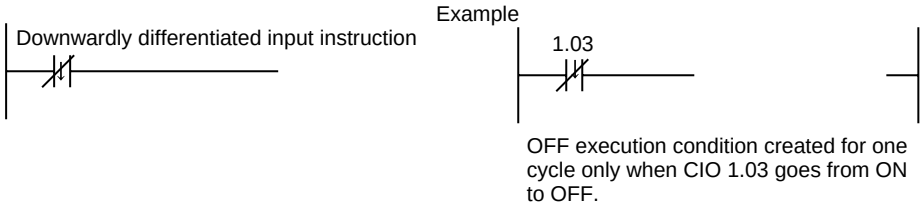


- **Input Instructions (Logical Starts and Intermediate Instructions)**: The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output the execution condition (power flow) when results switch from ON to OFF. The execution condition will turn OFF the next cycle.



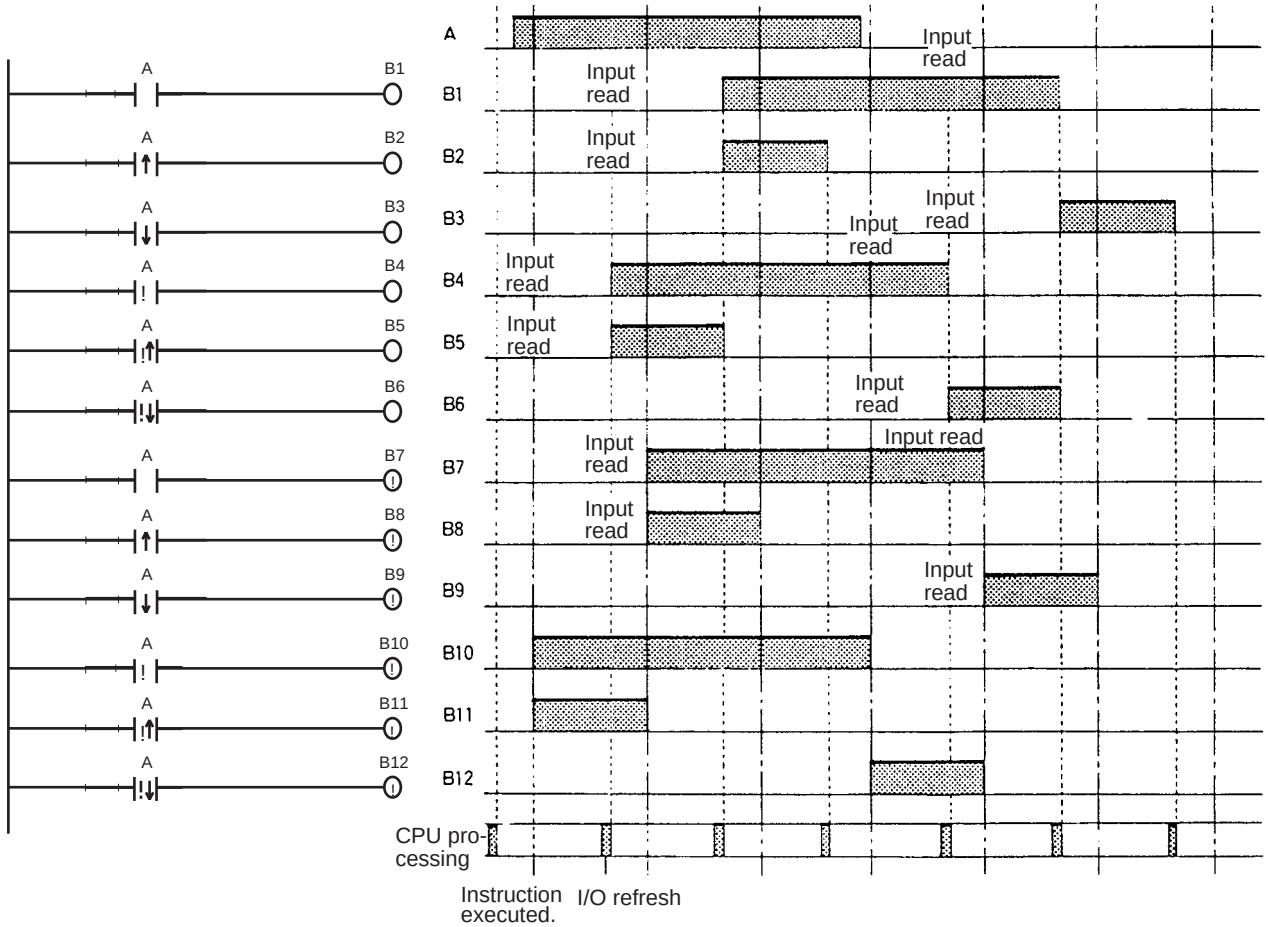
Note Unlike the upwardly differentiated instructions, downward differentiation variation (%) can only be added to LD, AND, OR, SET and RSET instructions. To execute downward differentiation with other instructions, combine the instructions with a DIFD or a DOWN instruction.

- **Input Instructions (Logical Starts and Intermediate Instructions)**: The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output an OFF execution condition (power flow stops) when results switch from ON to OFF. The execution condition will turn ON the next cycle.



1-1-9 I/O Instruction Timing

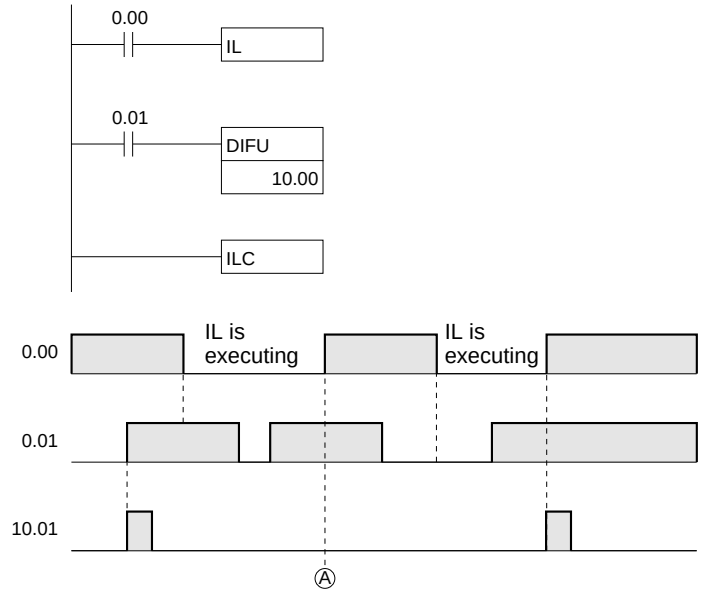
The following timing chart shows different operating timing for individual instructions using a program comprised of only LD and OUT instructions.



Differentiated Instructions

- A differentiated instruction has an internal flag that tells whether the previous value is ON or OFF. At the start of operation, the previous value flags for upwardly differentiated instruction (DIFU and @ instructions) are set to ON and the previous value flags for downwardly differentiated instructions (DIFD and % instructions) are set to OFF. This prevents differentiation outputs from being output unexpectedly at the start of operation.

- An upwardly differentiated instruction (DIFU or @ instruction) will output ON only when the execution condition is ON and flag for the previous value is OFF.
- **Use in Interlocks (IL - ILC Instructions)**
  In the following example, the previous value flag for the differentiated instruction maintains the previous interlocked value and will not output a differentiated output at point A because the value will not be updated while the interlock is in effect.



- **Use in Jumps (JMP - JME Instructions):** Just as for interlocks, the previous value flag for a differentiated instruction is not changed when the instruction is jumped, i.e., the previous value is maintained. Upwardly and downwardly differentiate instructions will output the execution condition only when the input status has changed from the status indicated by the previous value flag.

Note (a) Do not use the Always ON Flag or A200.11 (First Cycle Flag) as the input bit for an upwardly differentiated instruction. The instruction will never be executed.

(b) Do not use Always OFF Flag as the input bit for a downwardly differentiated instruction. The instruction will never be executed.

1-1-10 Refresh Timing

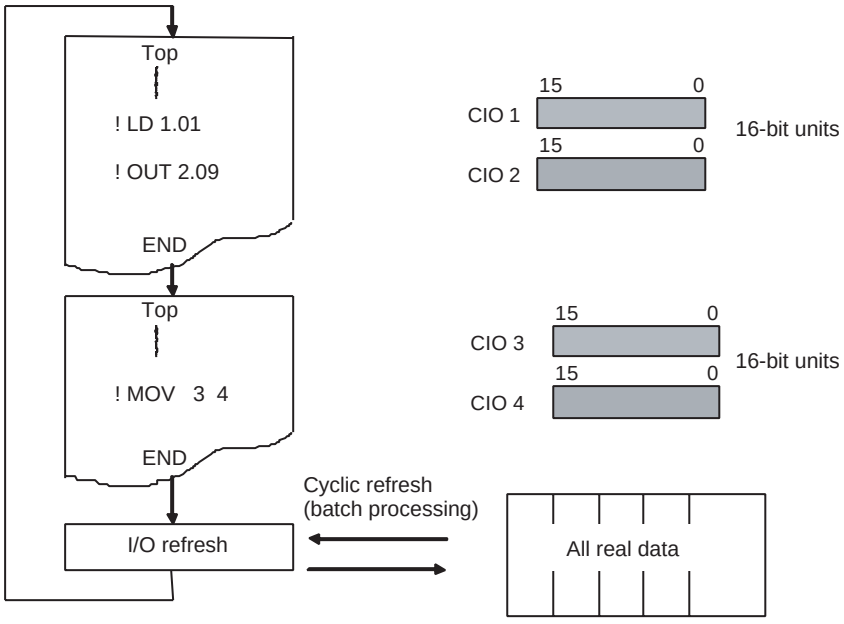
The following methods are used to refresh external I/O.

- Cyclic refresh
- Immediate refresh (! specified instruction, IORF instruction)

Cyclic Refresh

Every program allocated to a ready cyclic task or a task where interrupt condition has been met will execute starting from the beginning program address and will run until the END(001) instruction. After all ready cyclic tasks or tasks where interrupt condition have been met have executed, cyclic refresh will refresh all I/O points at the same time.

Note Programs can be executed in multiple tasks. I/O will be refreshed after the final END(001) instruction in the program allocated to the highest number (among all ready cyclic tasks) and will not be refreshed after the END(001) instruction in programs allocated to other cyclic tasks.



Execute IORF(097) for all required words or use instructions with the immediate refresh option prior to the END(001) instruction if I/O refreshing is required in other tasks.

Immediate Refresh

Instructions with Refresh Variation (!)

I/O will be refreshed as shown below when an instruction is executing if an real I/O bit in the built-in I/O of the CPU Unit is specified as an operand.

- When a bit operand is specified for an instruction, I/O will be refreshed for the 16 bits of the word containing the bit.
- When a word operand is specified for an instruction, I/O will be refreshed for the 16 bits that are specified.
- Inputs will be refreshed for input or source operand just before an instruction is executed.
- Outputs will be refreshed for outputs or destination (D) operands just after an instruction is execute.

Add an exclamation mark (!) (immediate refresh option) in front of the instruction.

Note Immediate refreshing is not supported for real I/O data allocated to CPM1A Expansion Units or Expansion I/O Units, but IORF(097) is supported for them.

Units Refreshed for IORF(097)

An I/O REFRESH instruction (IORF(097)) that refreshes real I/O data in a specified word range is available as a special instruction for CPM1A Expansion Units and Expansion I/O Units. All or just a specified range of real I/O bits can be refreshed during a cycle with this instruction.

Note IORF(097) cannot be used for real I/O bits allocated to the built-in I/O of the CPU Unit. Use instructions with the immediate refresh option for this I/O.

IORF(097) can also be used to refresh words allocated to CJ-series Special I/O Units.

DLNK(226)

The CPU BUS UNIT I/O REFRESH instruction (DLNK(226)) can be used to refresh memory allocated to CJ-series CPU Bus Units in the CIO and DM Areas, as well as data link data and other data specific to the CPU Bus Units. The unit number of the CPU Bus Unit is specified when DLNK(226) is executed to refresh all of the following data at the same time.

- Words allocated to the Unit in CIO Area
- Words allocated to the Unit in DM Area
- Special refreshing for the Unit (e.g., data links for Controller Link Units or remote I/O for DeviceNet Units)

1-1-11 Program Capacity

The maximum program capacities of the CP-series CPU Units for all user programs (i.e., the total capacity of all tasks) are given in the following table. All capacities are given as the maximum number of steps. The capacities must not be exceeded, and writing the program will be disabled if an attempt is made to exceed the capacity.

Each instruction is from 1 to 7 steps long. Refer to *SECTION 4 Instruction Execution Times and Number of Steps* for the specific number of steps in each instruction. (The length of each instruction will increase by 1 step if a double-length operand is used.)

Series	CPU Unit type	Model	Max. program capacity
CP Series CP1H CPU Units	XA	CP1H-XA40D□-□	20K steps
	X	CP1H-X40D□-□	
	Y	CP1H-Y20DT-D	

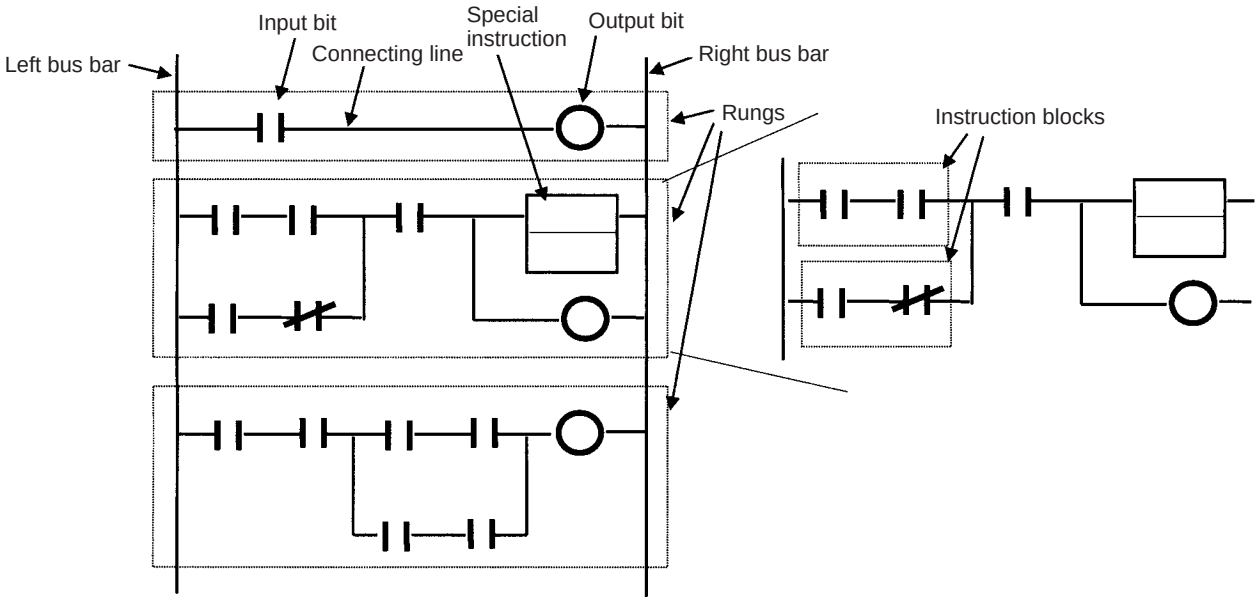
Note Memory capacity for CP-series PLCs is measured in steps, whereas memory capacity for previous OMRON PLCs, such as the C200HX/HG/HE and CV-series PLCs, was measured in words. Refer to the information at the end of *SECTION 4 Instruction Execution Times and Number of Steps* for guidelines on converting program capacities from previous OMRON PLCs.

1-1-12 Basic Ladder Programming Concepts

Instructions are executed in the order listed in memory (mnemonic order). The basic programming concepts as well as the execution order must be correct.

General Structure of the Ladder Diagram

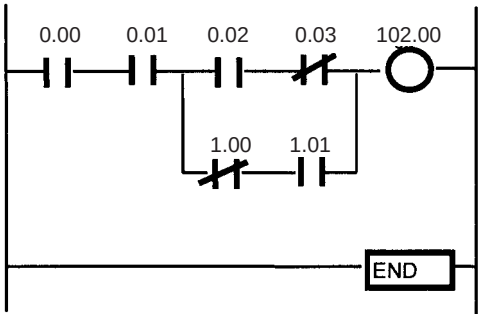
A ladder diagram consists of left and right bus bars, connecting lines, input bits, output bits, and special instructions. A program consists of one or more program runs. A program rung is a unit that can be partitioned when the bus is split horizontally. In mnemonic form, a rung is all instructions from a LD/LD NOT instruction to the output instruction just before the next LD/LD NOT instructions. A program rung consists of instruction blocks that begin with an LD/LD NOT instruction indicating a logical start.



Mnemonics

A mnemonic program is a series of ladder diagram instructions given in their mnemonic form. It has program addresses, and one program address is equivalent to one instruction.

Example



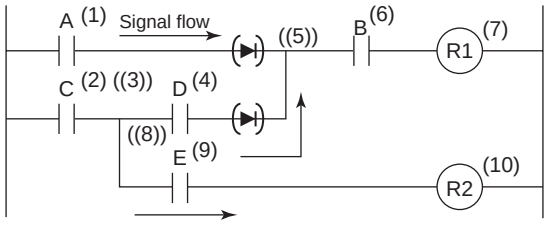
Program Address	Instruction (Mnemonic)	Operand
0	LD	0.00
1	AND	0.01
2	LD	0.02
3	AND NOT	0.03
4	LD NOT	1.00
5	AND	1.01
6	OR LD	
7	AND LD	
8	OUT	102.00
9	END	

Basic Ladder Program Concepts

1,2,3... 1. When ladder diagrams are executed by PLCs, the signal flow (power flow) is always from left to right. Programming that requires power flow from right to left cannot be used. Thus, flow is different from when circuits are made up of hard-wired control relays. For example, when the circuit “a” is implemented in a PLC program, power flows as though the diodes in brackets

were inserted and coil R2 cannot be driven with contact D included. The actual order of execution is indicated on the right with mnemonics. To achieve operation without these imaginary diodes, the circuit must be re-written. Also, circuit “b” power flow cannot be programmed directly and must be rewritten.

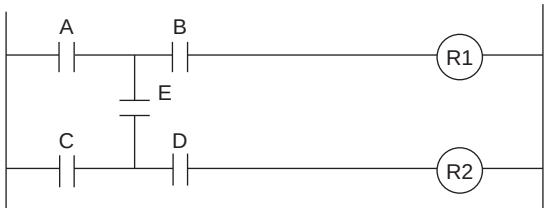
Circuit "a"



Order of execution (mnemonic)

(1) LD A	(6) AND B
(2) LD C	(7) OUT R1
(3) OUT TR0	(8) LD TR0
(4) AND D	(9) AND E
(5) OR LD	(10) OUT R2

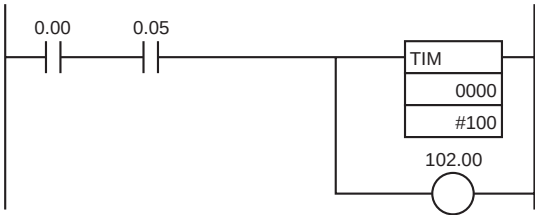
Circuit " b"



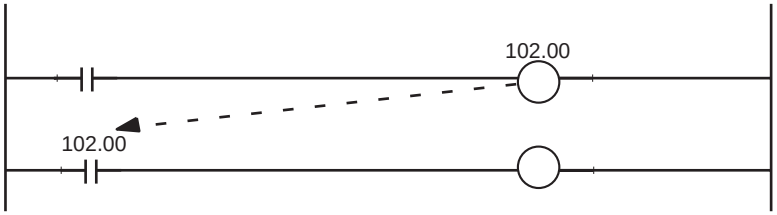
In circuit “a,” coil R2 cannot be driven with contact D included.

In circuit “b,” contact E included cannot be written in a ladder diagram. The program must be rewritten.

2. There is no limit to the number of I/O bits, work bits, timers, and other input bits that can be used. Rungs, however, should be kept as clear and simple as possible even if it means using more input bits to make them easier to understand and maintain.
3. There is no limit to the number of input bits that can be connected in series or in parallel in series or parallel rungs.
4. Two or more output bits can be connected in parallel.



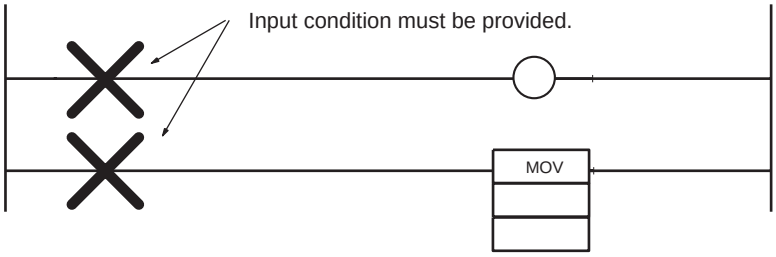
5. Output bits can also be used as input bits.



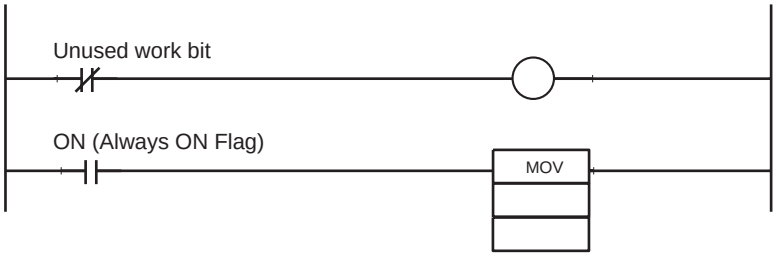
Restrictions

1,2,3... 1. A ladder program must be closed so that signals (power flow) will flow from the left bus bar to the right bus bar. A rung error will occur if the program is not closed (but the program can be executed).

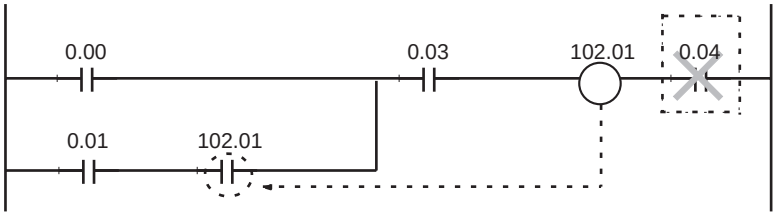
2. Output bits, timers, counters and other output instructions cannot be connected directly to the left bus bar. If one is connected directly to the left bus bar, a rung error will occur during the programming check by the CX-Programmer. (The program can be executed, but the OUT and MOV(021) will not be executed.)



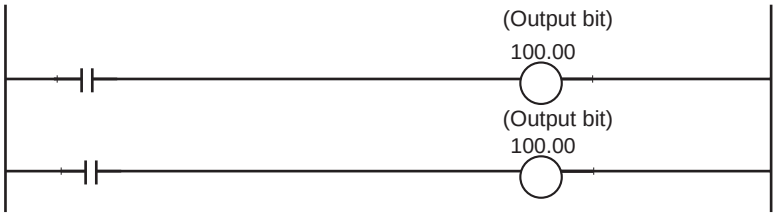
Insert an unused N.C. work bit or the ON Condition Flag (Always ON Flag) if the input must be kept ON at all times.



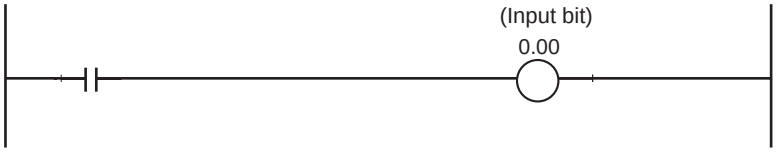
3. An input bit must always be inserted before and never after an output instruction like an output bit. If it is inserted after an output instruction, then a location error will occur during the CX-Programmer program check.



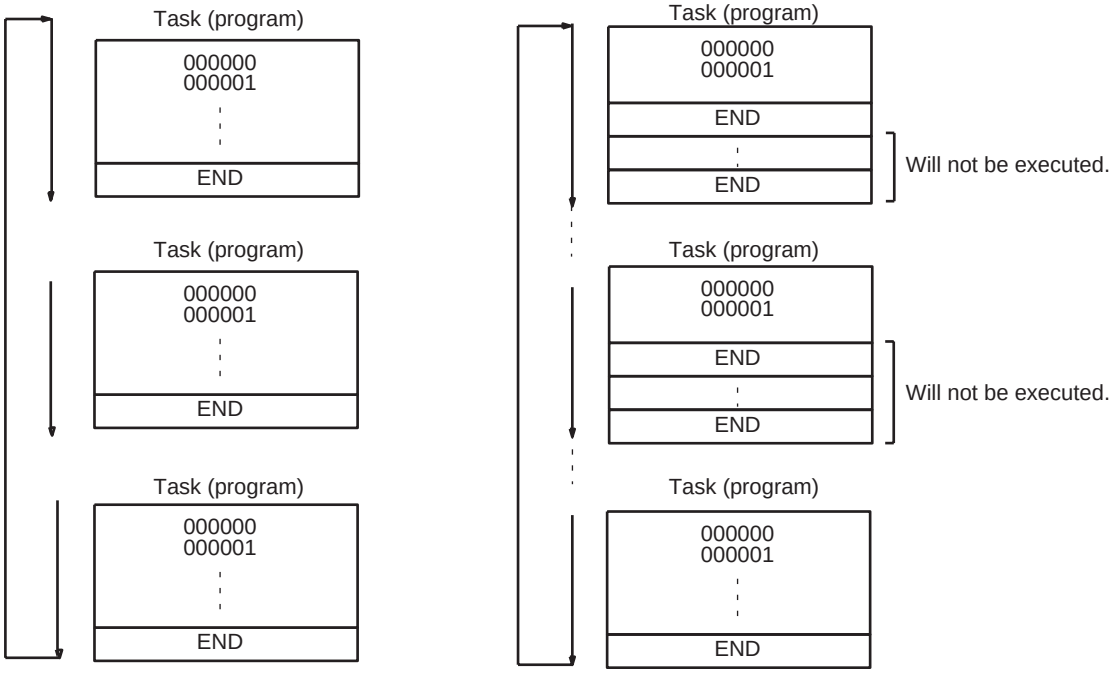
4. The same output bit cannot be programmed in an output instruction more than once. Instructions in a ladder program are executed in order from the top rung in a single cycle, so the result of output instruction in the lower rungs will be ultimately reflected in the output bit and the results of any previous instructions controlling the same bit will be overwritten and not output.



5. An input bit cannot be used in an OUTPUT instruction (OUT).



6. An END(001) instruction must be inserted at the end of the program in each task.
   - If a program without an END(001) instruction starts running, a program error indicating No End Instruction will occur, the ERR/ALM LED on the front of the CPU Unit will light, and the program will not be executed.
   - If a program has more than one END(001) instruction, then the program will only run until the first END(001) instruction.
   - Debugging programs will run much smoother if an END(001) instruction is inserted at various break points between sequence rungs and the END(001) instruction in the middle is deleted after the program is checked.

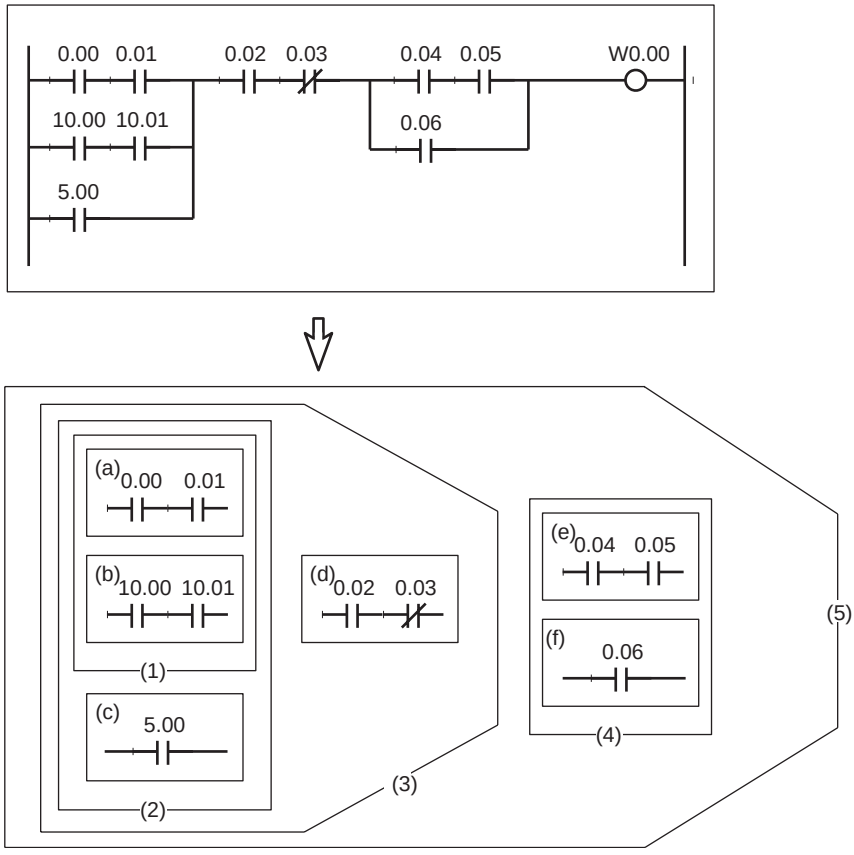


1-1-13 Inputting Mnemonics

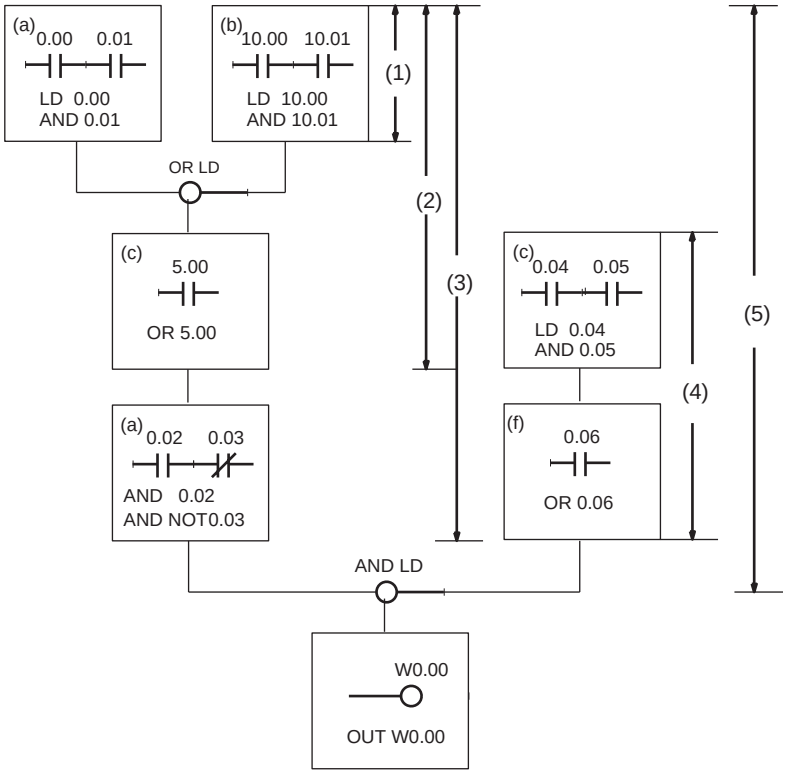
A logical start is accomplished using an LD/LD NOT instruction. The area from the logical start until the instruction just before the next LD/LD NOT instruction is considered a single instruction block.

Create a single rung consisting of two instruction blocks using an AND LD instruction to AND the blocks or by using an OR LD instruction to OR the blocks. The following example shows a complex rung that will be used to explain the procedure for inputting mnemonics (rung summary and order).

1,2,3... 1. First separate the rung into small blocks (a) to (f).



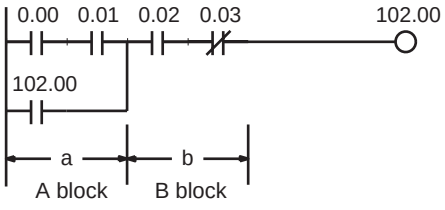
2. Program the blocks from top to bottom and then from left to right.



	Address	Instruction	Operand
(a)	200	LD	0.00
	201	AND	0.01
(b)	202	LD	10.00
	203	AND	10.01
	204	OR LD	---
(c)	205	OR	5.00
(d)	206	AND	0.02
	207	AND NOT	0.03
(e)	208	LD	0.04
	209	AND	0.05
(f)	210	OR	0.06
	211	AND LD	---
	212	OUT	W0.00

1-1-14 Program Examples

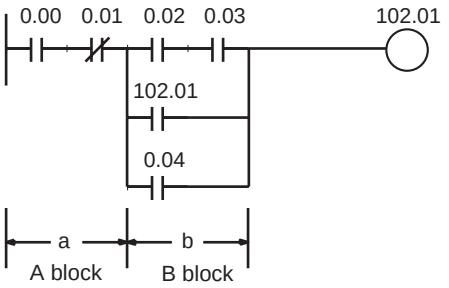
Parallel/Series Rungs



Instruction	Operands	
LD	0.00	a
AND	0.01	
OR	102.00	
AND	0.02	b
AND NOT	0.03	
OUT	102.00	

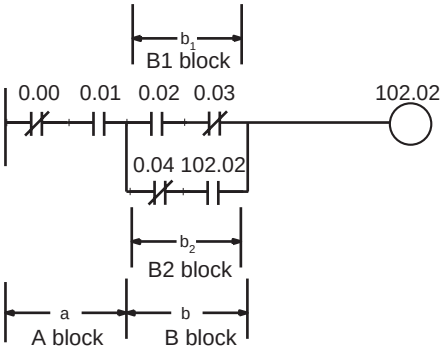
Program the parallel instruction in the A block and then the B block.

Series/Parallel Rungs



Instruction	Operands	
LD	0.00	a
AND NOT	0.01	
LD	0.02	b
AND	0.03	
OR	102.01	
OR	0.04	
AND LD	---	
OUT	102.01	

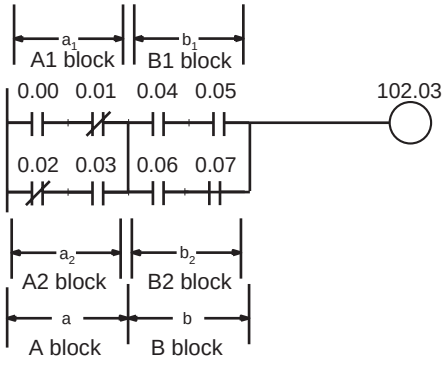
- Separate the rung into A and B blocks, and program each individually.
- Connect A and B blocks with an AND LD.
- Program A block.



Instruction	Operands	
LD NOT	0.00	a
AND	0.01	
LD	0.02	b_1
AND NOT	0.03	
LD NOT	0.04	b_2
AND	102.02	
OR LD	---	$b_1 + b_2$
AND LD	---	$a \cdot b$
OUT	102.02	

- Program B_1 block and then program B_2 block.
- Connect B_1 and B_2 blocks with an OR LD and then A and B blocks with an AND LD.

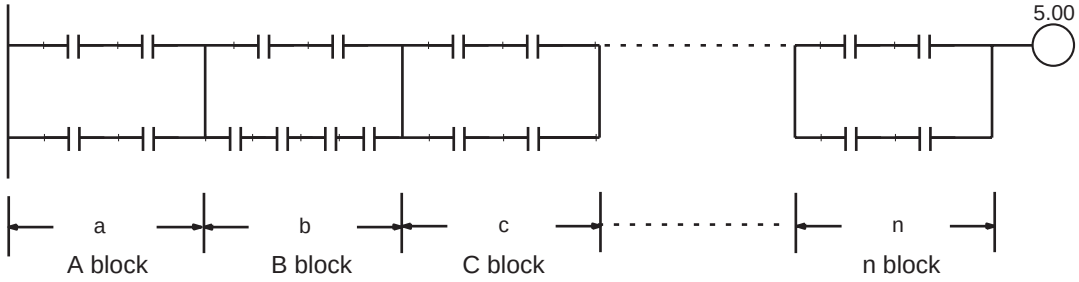
Example of Series Connection in a Series Rung



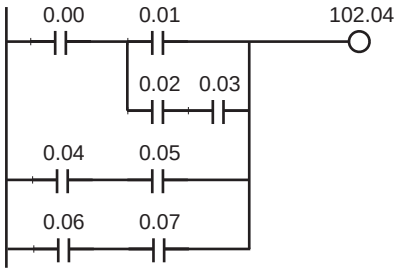
Instruction	Operands	
LD	0.00	a_1
AND NOT	0.01	
LD NOT	0.02	a_2
AND	0.03	
OR LD	---	$a_1 + a_2$
LD	0.04	b_1
AND	0.05	
LD	0.06	b_2
AND	0.07	
OR LD	---	$b_1 + b_2$
AND LD	---	$a \cdot b$
OUT	102.03	

- Program A_1 block, program A_2 block, and then connect A_1 and A_2 blocks with an OR LD.
- Program B_1 and B_2 the same way.
- Connect A block and B block with an AND LD.

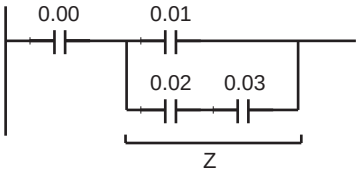
• Repeat for as many A to n blocks as are present.



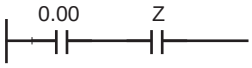
Complex Rungs



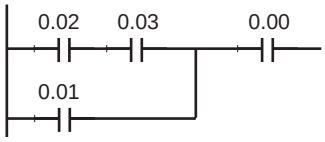
Instruction	Operand	
LD	0.00	
LD	0.01	
LD	0.02	
AND	0.03	
OR LD	---	Z
AND LD	---	
LD	0.04	
AND	0.05	
OR LD	---	
LD	0.06	
AND	0.07	
OR LD	---	
OUT	102.04	

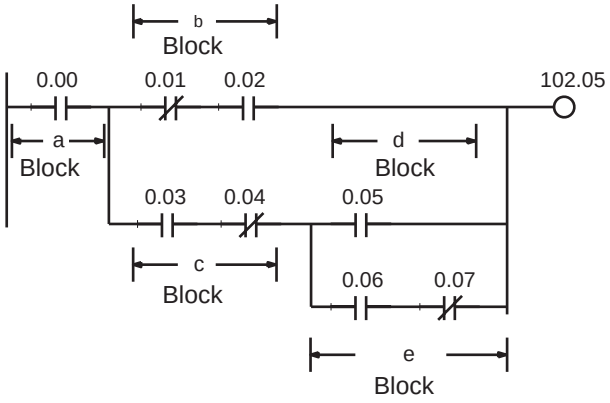


The diagram above is based on the diagram below.



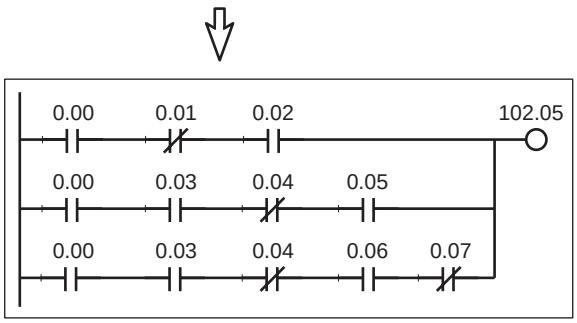
A simpler program can be written by rewriting this as shown below.

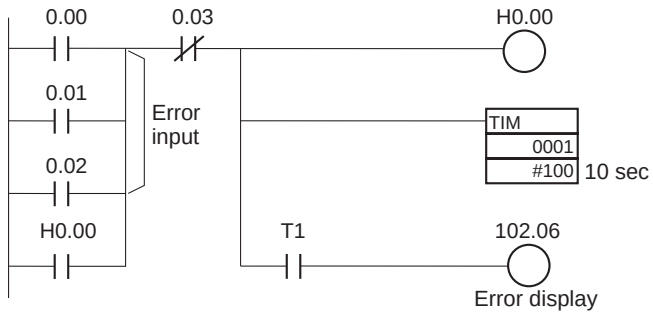




Instruction	Operand	
LD	0.00	a
LD NOT	0.01	b
AND	0.02	b
LD	0.03	c
AND NOT	0.04	c
LD	0.05	d
LD	0.06	d
AND NOT	0.07	e
OR LD	---	e
AND LD	---	d + e
OR LD	---	(d + e) · c
AND LD	---	(d + e) · c + b
OUT	102.05	((d + e) · c + b) · a

The above rung can be rewritten as follows:





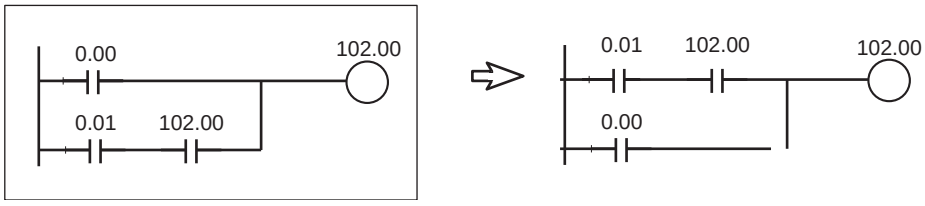
Instruction	Operand
LD	0.00
OR	0.01
OR	0.02
OR	H0.00
AND NOT	0.03
OUT	H0.00
TIM	0001
	#100
AND	T1
OUT	102.06

If a holding bit is in use, the ON/OFF status would be held in memory even if the power is turned OFF, and the error signal would still be in effect when power is turned back ON.

Rungs Requiring Caution or Rewriting

OR and OL LD Instructions

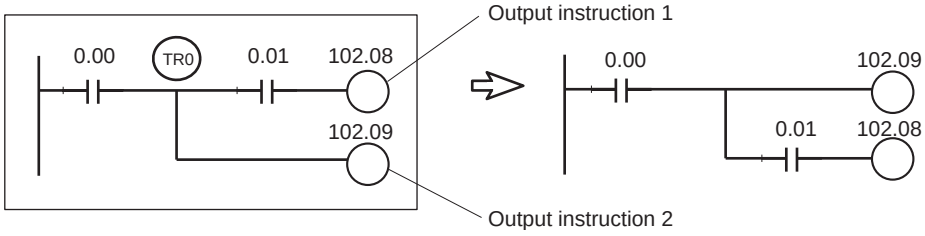
With an OR or OR NOT instruction, an OR is taken with the results of the ladder logic from the LD or LD NOT instruction to the OR or OR NOT instruction, so the rungs can be rewritten so that the OR LD instruction is not required.



Example: An OR LD instruction will be needed if the rungs are programmed as shown without modification. A few steps can be eliminated by rewriting the rungs as shown.

Output Instruction Branches

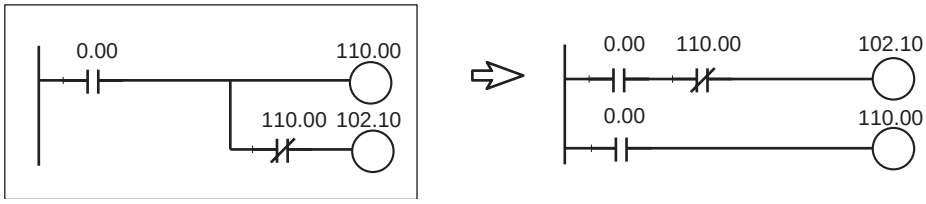
A TR bit will be needed if there is a branch before an AND or AND NOT instruction. The TR bit will not be needed if the branch comes at a point that is connected directly to output instructions and the AND or AND NOT instruction or the output instructions can be continued as is.



Example: A temporary storage bit TR0 output instruction and load (LD) instruction are needed at a branch point if the rungs are programmed without modification. A few steps can be eliminated by rewriting the rungs.

Mnemonic Execution Order

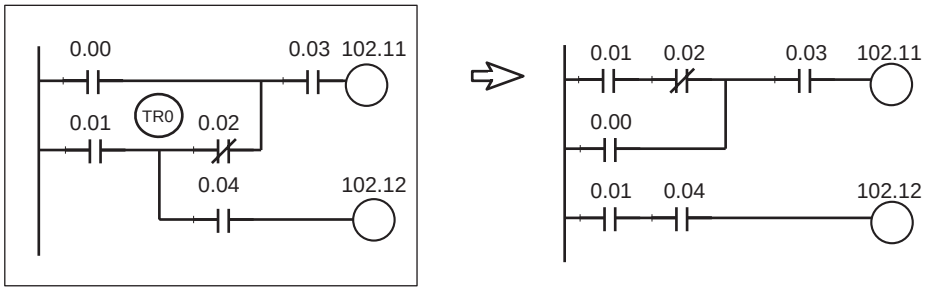
PLCs execute ladder programs in the order the mnemonics are entered so instructions may not operate as expected, depending on the way rungs are written. Always consider mnemonic execution order when writing ladder diagrams.



Example: CIO 102.10 in the above diagram cannot be output. By rewriting the rung, as shown above, CIO 102.10 can be turned ON for one cycle.

Rungs Requiring Rewriting

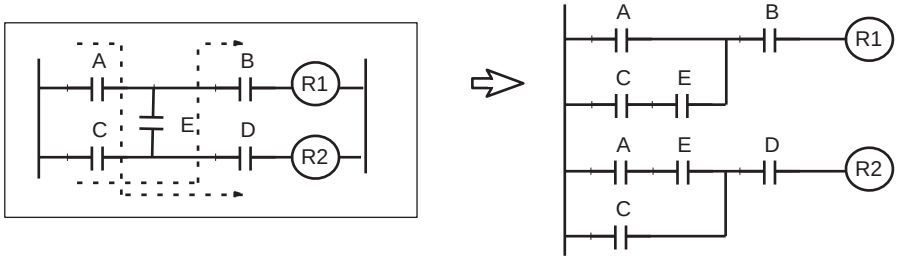
PLCs execute instructions in the order the mnemonics are entered so the signal flow (power flow) is from left to right in the ladder diagram. Power flows from right to left cannot be programmed.



Example: The program can be written as shown in the diagram at the left where TR0 receives the branch. The same value is obtained, however, by the rungs at the right, which are easier to understand. It is recommended, therefore, that the rungs at the left be rewritten to the rungs at the right.

Rewrite the rungs on the left below. They cannot be executed.

The arrows show signal flow (power flow) when the rungs consist of control relays.



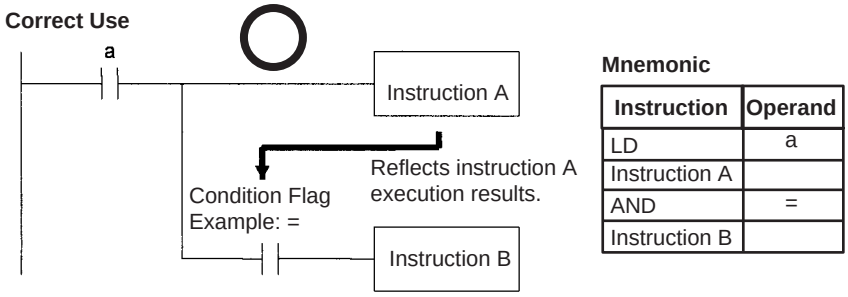
1-2 Precautions

1-2-1 Condition Flags

Using Condition Flags

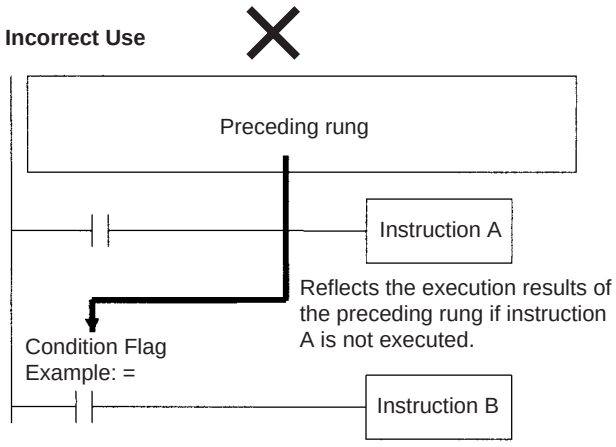
Conditions flags are shared by all instructions, and will change during a cycle depending on results of executing individual instructions. Therefore, be sure to use Condition Flags on a branched output with the same execution condition immediately after an instruction to reflect the results of instruction execution. Never connect a Condition Flag directly to the bus bar because this will cause it to reflect execution results for other instructions.

Example: Using Instruction A Execution Results



Instruction	Operand
LD	a
Instruction A	
AND	=
Instruction B	

The same execution condition (a) is used for instructions A and B to execute instruction B based on the execution results of instruction A. In this case, instruction B will be executed according to the Condition Flag only if instruction A is executed.

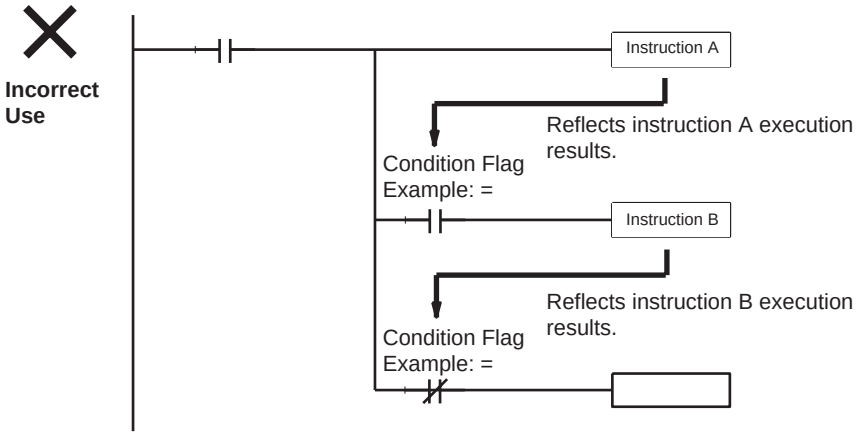


If the Condition Flag is connected directly to the left bus bar, instruction B will be executed based on the execution results of a previous rung if instruction A is not executed.

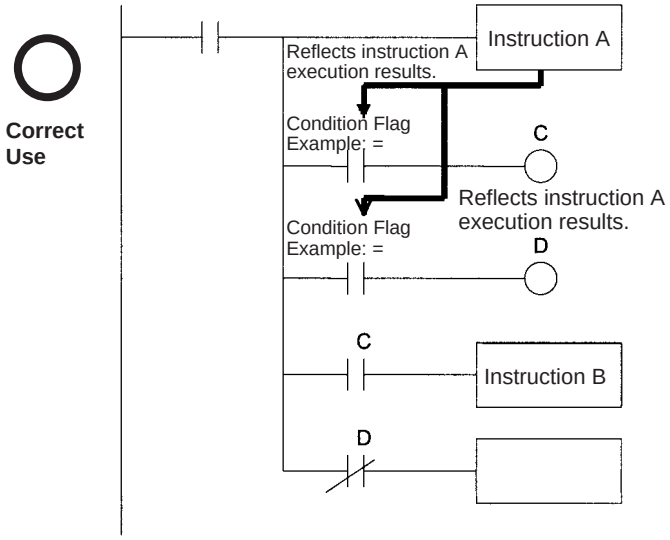
Note Condition Flags are used by all instruction within a single program (task) but they are cleared when the task switches. Therefore execution results in the preceding task will not be reflected later tasks. Since conditions flags are shared by all instructions, make absolutely sure that they do not interfere with each other within a single ladder-diagram program. The following is an example.

Using Execution Results in N.C. and N.C. Inputs

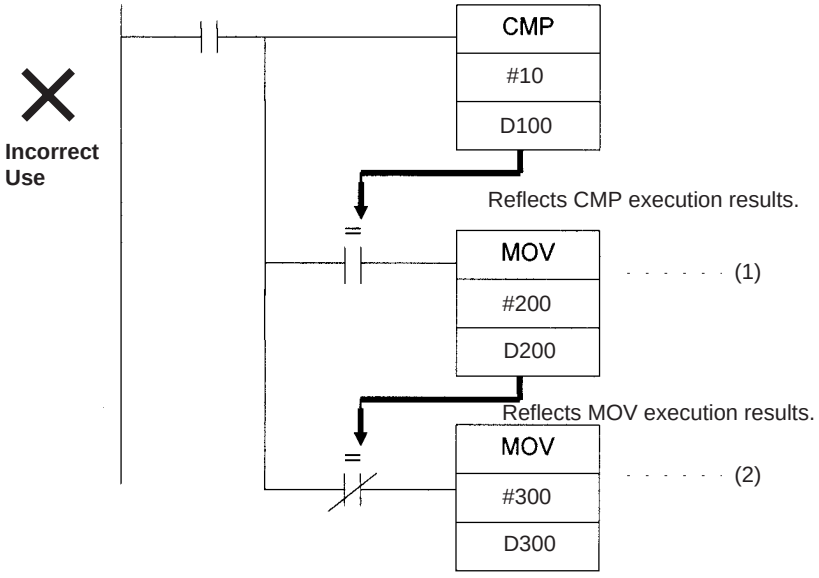
The Condition Flags will pick up instruction B execution results as shown in the example below even though the N.C. and N.O. input bits are executed from the same output branch.



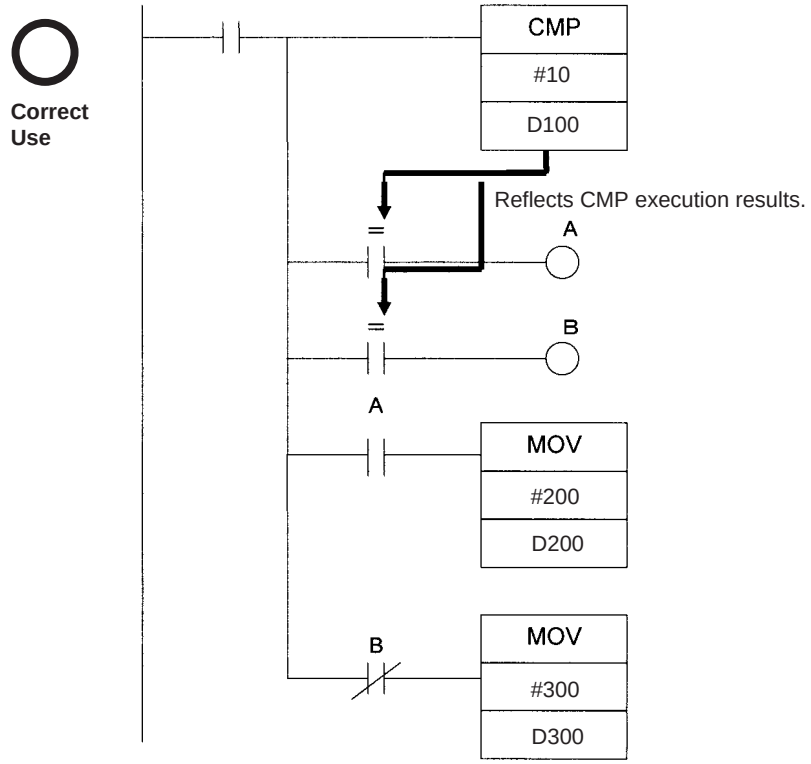
Make sure each of the results is picked up once by an OUTPUT instruction to ensure that execution results for instruction B will be not be picked up.



Example: The following example will move #200 to D200 if D100 contains #10 and move #300 to D300 if D100 does not contain #10.



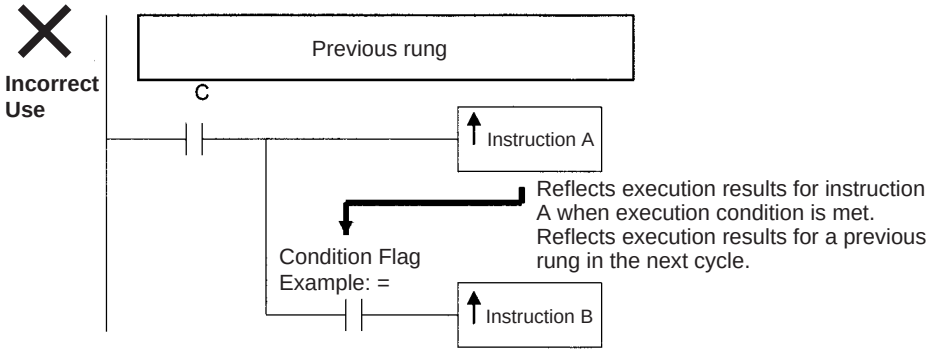
The Equals Flag will turn ON if D100 in the rung above contains #10. #200 will be moved to D200 for instruction (1), but then the Equals Flag will be turned OFF because the #200 source data is not 0000 hex. The MOV instruction at (2) will then be executed and #300 will be moved to D300. A rung will therefore have to be inserted as shown below to prevent execution results for the first MOVE instruction from being picked up.



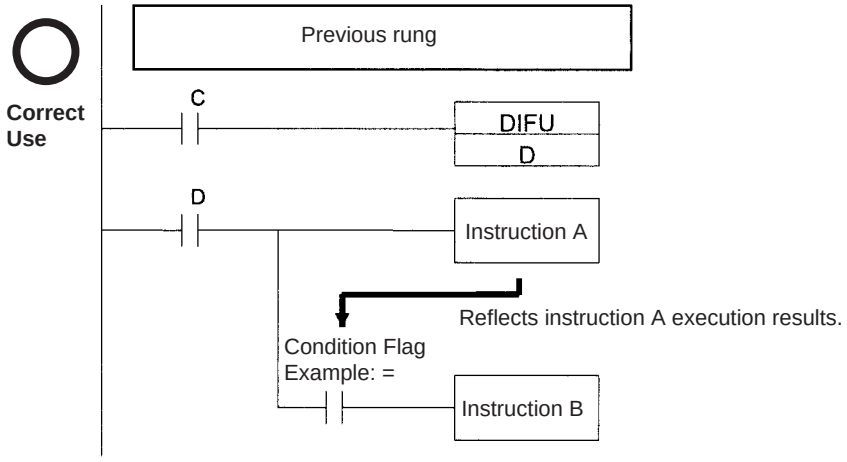
Using Execution Results from Differentiated Instructions

With differentiated instructions, execution results for instructions are reflected in Condition Flags only when execution condition is met, and results for a previous rung (rather than execution results for the differentiated instruction) will be reflected in Condition Flags in the next cycle. You must therefore be aware of what Condition Flags will do in the next cycle if execution results for differentiated instructions to be used.

In the following for example, instructions A and B will execute only if execution condition C is met, but the following problem will occur when instruction B picks up execution results from instruction A. If execution condition C remains ON in the next cycle after instruction A was executed, then instruction B will unexpectedly execute (by the execution condition) when the Condition Flag goes from OFF to ON because of results reflected from a previous rung.



In this case then, instructions A and B are not differentiated instructions, the DIFU (of DIFD) instruction is used instead as shown below and instructions A and B are both upwardly (or downwardly) differentiated and executed for one cycle only.



Note The CP1H CPU Units support instructions to save and load the Condition Flag status (CCS(282) and CCL(283)). These can be used to access the status of the Condition Flags at other locations in a task or in a different task.

Main Conditions Turning ON Condition Flags

Error Flag

The ER Flag will turn ON under special conditions, such as when operand data for an instruction is incorrect. The instruction will not be executed when the ER Flag turns ON.

When the ER Flag is ON, the status of other Condition Flags, such as the <, >, OF, and UF Flags, will not change and status of the = and N Flags will vary from instruction to instruction.

Refer to the descriptions of individual instructions in the *CP-series CP1H CPU Unit Programming Manual (W451)* for the conditions that will cause the ER Flag to turn ON. Caution is required because some instructions will turn OFF the ER Flag regardless of conditions.

Note The PLC Setup Settings for when an instruction error occurs determines whether operation will stop when the ER Flag turns ON. In the default setting, operation will continue when the ER Flag turns ON. If Stop Operation is specified when the ER Flag turns ON and operation stops (treated as a program error), the program address at the point where operation stopped will be stored at in A298 to A299. At the same time, A295.08 will turn ON.

Equals Flag

The Equals Flag is a temporary flag for all instructions except when comparison results are equal (=). It is set automatically by the system, and it will change. The Equals Flag can be turned OFF (ON) by an instruction after a previous instruction has turned it ON (OFF). The Equals Flag will turn ON, for example, when MOV or another move instruction moves 0000 hex as source data and will be OFF at all other times. Even if an instruction turns the Equals Flag ON, the move instruction will execute immediately and the Equals Flag will turn ON or OFF depending on whether the source data for the move instruction is 0000 hex or not.

Carry Flag

The CY Flag is used in shift instructions, addition and subtraction instructions with carry input, addition and subtraction instruction borrows and carries, as well as with Special I/O Unit instructions, PID instructions, and FPD instructions. Note the following precautions.

Note

(1) The CY Flag can remain ON (OFF) because of execution results for a certain instruction and then be used in other instruction (an addition and subtraction instruction with carry or a shift instruction). Be sure to clear the Carry Flag when necessary.

(2) The CY Flag can be turned ON (OFF) by the execution results for a certain instruction and be turned OFF (ON) by another instruction. Be sure the proper results are reflected in the Carry Flag when using it.

Less Than and Greater Than Flags

The < and > Flags are used in comparison instruction, as well as in the LMT, BAND, ZONE, PID and other instructions.
The < or > Flag can be turned OFF (ON) by another instruction even if it is turned ON (OFF) by execution results for a certain instruction.

Negative Flag

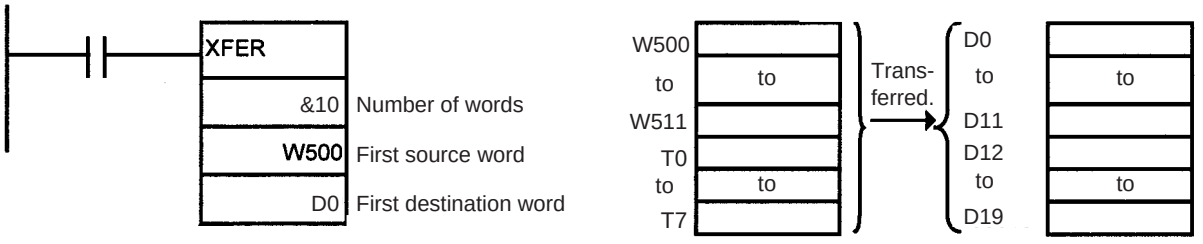
The N Flag is turned OFF when the leftmost bit of the instruction execution results word is “1” for certain instructions and it is turned OFF unconditionally for other instruction.

Specifying Operands for Multiple Words

With the CP-series PLCs, an instruction will be executed as written even if an operand requiring multiple words is specified so that all of the words for the operand are not in the same area. In this case, words will be taken in order of the PLC memory addresses. The Error Flag will **not** turn ON.

As an example, consider the results of executing a block transfer with XFER(070) if 20 words are specified for transfer beginning with W500. Here, the Work Area, which ends at W511, will be exceeded, but the instruction will be executed without turning ON the Error Flag. In the PLC memory addresses, the present values for timers are held in memory after the Work Area, and thus for the following instruction, W500 to W511 will be transferred to D0 to D11 and the present values for T0 to T7 will be transferred to D12 to D19.

Note Refer to the appendix *Memory Map of PLC Memory Addresses* in the *CP-series CP1H CPU Unit Operation Manual (W450)* for specific PLC memory addresses.



1-2-2 Special Program Sections

CP-series programs have special program sections that will control instruction conditions. The following special program sections are available.

Program section	Instructions	Instruction condition	Status
Subroutine	SBS, SBN and RET instructions	Subroutine program is executed.	The subroutine program section between SBN and RET instructions is executed.
IL - ILC section	IL and ILC instructions	Section is interlocked	The output bits are turned OFF and timers are reset. Other instructions will not be executed and previous status will be maintained.
Step Ladder section	STEP S instructions and STEP instructions		
FOR-NEXT loop	FOR instructions and NEXT instructions	Break in progress.	Looping
JMP0 - JME0 section	JMP0 instructions and JME0 instructions		Jump
Block program section	BPRG instructions and BEND instructions	Block program is executing.	The block program listed in mnemonics between the BPRG and BEND instructions is executed.

Instruction Combinations

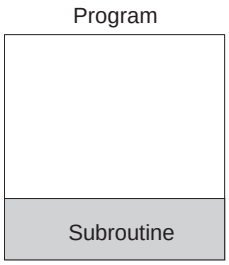
The following table shows which of the special instructions can be used inside other program sections.

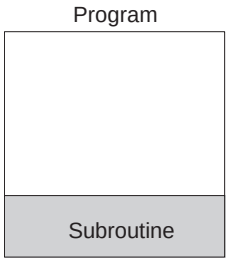
	Subroutine	IL - ILC section	Step ladder section	FOR - NEXT loop	JMP0 - JME0 section	Block program section
Subroutine	Not possible.	Not possible.	Not possible.	Not possible.	Not possible.	Not possible.
IL - ILC	**OK**	Not possible.	Not possible.	**OK**	**OK**	Not possible.
Step ladder section	Not possible.	**OK**	Not possible.	Not possible.	**OK**	Not possible.
FOR - NEXT loop	**OK**	**OK**	Not possible.	**OK**	**OK**	Not possible.
JMP0 - JME0	**OK**	**OK**	Not possible.	Not possible.	Not possible.	Not possible.
Block program section	**OK**	**OK**	**OK**	Not possible.	**OK**	Not possible.

Note Instructions that specify program areas cannot be used for programs in other tasks. Refer to *2-2-2 Task Instruction Limitations* for details.

Subroutines

Place all the subroutines together just before the END(001) instruction in all programs but after programming other than subroutines. (Therefore, a subroutine cannot be placed in a step ladder, block program, FOR - NEXT, or JMP0 - JME0 section.) If a program other than a subroutine program is placed after a subroutine program (SBN to RET), that program will not be executed.





Instructions Not Available in Subroutines

The following instructions cannot be placed in a subroutine.

Function	Mnemonic	Instruction
Process Step Control	STEP(008)	Define step ladder section
	SNXT(009)	Step through the step ladder

Note **Block Program Sections**
A subroutine can include a block program section. If, however, the block program is in WAIT status when execution returns from the subroutine to the main program, the block program section will remain in WAIT status the next time it is called.

Instructions Not Available in Step Ladder Program Sections

Function	Mnemonic	Instruction
Sequence Control	FOR(512), NEXT(513), and BREAK(514)	FOR, NEXT, and BREAK LOOP
	END(001)	END
	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR
	JMP(004) and JME(005)	JUMP and JUMP END
	CJP(510) and CJPN(511)	CONDITIONAL JUMP and CONDITIONAL JUMP NOT
	JMP0(515) and JME0(516)	MULTIPLE JUMP and MULTIPLE JUMP END
Subroutines	SBN(092) and RET(093)	SUBROUTINE ENTRY and SUBROUTINE RETURN

Function	Mnemonic	Instruction
Block Programs	IF(802) (NOT), ELSE(803), and IEND(804)	Branching instructions
	BPRG(096) and BEND(801)	BLOCK PROGRAM BEGIN/END
	EXIT(806) (NOT)	CONDITIONAL BLOCK EXIT (NOT)
	LOOP(809) and LEND(810) (NOT)	Loop control
	WAIT(805) (NOT)	ONE CYCLE WAIT (NOT)
	TIMW(813) and TIMWX(816)	TIMER WAIT
	TMHW(815) and TMHWX(817)	HIGH-SPEED TIMER WAIT
	CNTW(814) and CNTWX(818)	COUNTER WAIT
	BPPS(811) and BPRS(812)	BLOCK PROGRAM PAUSE and RESTART

Note

(1) A step ladder program section can be used in an interlock section (between IL and ILC). The step ladder section will be completely reset when the interlock is ON.

(2) A step ladder program section can be used between MULTIPLE JUMP (JMP0) and MULTIPLE JUMP END (JME0).

Instructions Not Supported in Block Program Sections

The following instructions cannot be placed in block program sections.

Classification by Function	Mnemonic	Instruction
Sequence Control	FOR(512), NEXT(513), and BREAK(514)	FOR, NEXT, and BREAK LOOP
	END(001)	END
	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR
	JMP0(515) and JME0(516)	MULTIPLE JUMP and MULTIPLE JUMP END
Sequence Input	UP(521)	CONDITION ON
	DOWN(522)	CONDITION OFF
Sequence Output	DIFU	DIFFERENTIATE UP
	DIFD	DIFFERENTIATE DOWN
	KEEP	KEEP
	OUT	OUTPUT
	OUT NOT	OUTPUT NOT
Timer/Counter	TIM and TIMX(550)	TIMER
	TIMH(015) and TIMHX(551)	HIGH-SPEED TIMER
	TMHH(540) and TMHHX(552)	ONE-MS TIMER
	TTIM(087) and TTIMX(555)	ACCUMULATIVE TIMER
	TIML(542) and TIMLX(553)	LONG TIMER
	MTIM(543) and MTIMX(554)	MULTI-OUTPUT TIMER
	CNT and CNTX(546)	COUNTER
	CNTR(012) and CNTRX(548)	REVERSIBLE COUNTER

Classification by Function	Mnemonic	Instruction
Subroutines	SBN(092) and RET(093)	SUBROUTINE ENTRY and SUBROUTINE RETURN
Data Shift	SFT	SHIFT
Ladder Step Control	STEP(008) and SNXT(009)	STEP DEFINE and STEP START
Data Control	PID	PID CONTROL
Block Program	BPRG(096)	BLOCK PROGRAM BEGIN
Damage Diagnosis	FPD(269)	FAILURE POINT DETECTION
Instructions with a differentiation option	@XXX	Instruction with upward differentiation
	%XXX	Instruction with downward differentiation

Note

(1) Block programs can be used in a step ladder program section.

(2) A block program can be used in an interlock section (between IL and ILC). The block program section will not be executed when the interlock is ON.

(3) A block program section can be used between MULTIPLE JUMP (JMP0) and MULTIPLE JUMP END (JME0).

(4) A JUMP instruction (JMP) and CONDITIONAL JUMP instruction (CJP/CJPN) can be used in a block program section. JUMP (JMP) and JUMP END (JME) instructions, as well as CONDITIONAL JUMP (CJP/CJPN) and JUMP END (JME) instructions cannot be used in the block program section unless they are used in pairs. The program will not execute properly unless these instructions are paired.

1-3 Checking Programs

CP-series programs can be checked at the following stages.

- Input check during CX-Programmer input and other operations
- Program check by CX-Programmer
- Instruction check during execution
- Fatal error check (program errors) during execution

1-3-1 CX-Programmer

The program will be automatically checked by the CX-Programmer at the following times.

Timing	Checked contents
When inputting ladder diagrams	Instruction inputs, operand inputs, programming patterns
When loading files	All operands for all instructions and all programming patterns
When downloading files	Models supported by the CP Series and all operands for all instructions
During online editing	Capacity, etc.

The results of checking are output to the text tab of the Output Window. Also, the left bus bar of illegal program sections will be displayed in red in ladder view.

1-3-2 Program Checks with the CX-Programmer

The errors that are detected by the program check provided by the CX-Programmer are listed in the following table.

The CX-Programmer does not check range errors for indirectly addressed operands in instructions. Indirect addressing errors will be detected in the program execution check and the ER Flag will turn ON, as described in the next section. Refer to individual instruction descriptions for details.

When the program is checked on the CX-Programmer, the operator can specify program check levels A, B, and C (in order of the seriousness of the error), as well as a custom check level.

Area	Check
Illegal data: Ladder diagramming	Instruction locations
	I/O lines
	Connections
	Instruction and operation completeness
Instruction support by PLC	Instructions and operands supported by PLC
	Instruction variations (NOT, !, @, and %)
	Object code integrity
Operand ranges	Operand area ranges
	Operand data types
	Access check for read-only words
	Operand range checks, including the following. • Constants (#, &, +, –) • Control codes • Area boundary checks for multi-word operands • Size relationship checks for multi-word operands • Operand range overlaps • Multi-word allocations • Double-length operands • Area boundary checks for offsets
Program capacity for PLC	Number of steps
	Overall capacity
	Number of tasks
Syntax	Call check for paired instructions • IL–ILC • JMP–JME, CJP/CJPN–JME • SBS–SBN–RET, MCRO–SBN–RET • STEP–SNXT • BPRG–BEND • IF–IEND • LOOP–LEND
	Restricted programming locations for BPRG–BEND
	Restricted programming locations for SBN–RET
	Restricted programming locations for STEP–SNXT
	Restricted programming locations for FOR–NEXT
	Restricted programming locations for interrupt tasks
	Required programming locations for BPRG–BEND
	Required programming locations for FOR–NEXT
	Illegal nesting
	END(001) instruction
	Number consistency
Ladder diagram structure	Stack overflows

Area	Check
Output duplication (See note.)	Duplicate output check • By bit • By word • Timer/counter instructions • Long words (2-word and 4-word) • Multiple allocated words • Start/end ranges • FAL numbers • Instructions with multiple output operands
Tasks	Check for tasks set for starting at beginning of operation
	Task program allocation

Note Output duplication is not checked between tasks, only within individual tasks.

Multi-word Operands

Memory area boundaries are checked for multi-word operands for the program check as shown in the following table.

Check items	The following functionality is provided by the CX-Programmer for multi-word operands that exceed a memory area boundary. • The program cannot be transferred to the CPU Unit. • The program also cannot be read from the CPU Unit. • Compiling errors are generated for the program check. • Warnings will appear on-screen during offline programming. • Warnings will appear on-screen during online editing in PROGRAM or MONITOR mode.

1-3-3 Program Execution Check

Operand and instruction location checks are performed on instructions during input and during program checks from the CX-Programmer. These are not, however, final checks.

The following checks are performed during instruction execution.

Type of error	Flag that turns ON for error	Stop/Continue operation
1. Instruction Processing Error	ER Flag **Note** The Instruction Processing Error Flag (A295.08) will also turn ON if Stop Operation is specified when an error occurs.	A setting in the PLC Setup can be used to specify whether to stop or continue operation for instruction processing errors. The default is to continue operation. A program error will be generated and operation will stop only if Stop Operation is specified.
2. Access Error	AER Flag **Note** The Access Error Flag (A295.10) will turn ON if Stop Operation is specified when an error occurs.	A setting in the PLC Setup can be used to specify whether to stop or continue operation for instruction processing errors. The default is to continue operation. A program error will be generated and operation will stop only if Stop Operation is specified.
3. Illegal Instruction Error	Illegal Instruction Error Flag (A295.14)	Fatal (program error)
4. UM (User Memory) Overflow Error	UM Overflow Error Flag (A295.15)	Fatal (program error)

Instruction Processing Errors

An instruction processing error will occur if incorrect data was provided when executing an instruction or an attempt was made to execute an instruction outside of a task. Here, data required at the beginning of instruction processing was checked and as a result, the instruction was not executed, the ER Flag (Error Flag) will be turned ON and the EQ and N Flags may be retained or turned OFF depending upon the instruction.

The ER Flag (error Flag) will turn OFF if the instruction (excluding input instructions) ends normally. Conditions that turn ON the ER Flag will vary with individual instructions. See descriptions of individual instructions in for details.

If Instruction Errors are set to Stop Operation in the PLC Setup, then operation will stop (fatal error) and the Instruction Processing Error Flag (A295.08) will turn ON if an instruction processing error occurs and the ER Flag turns ON.

Illegal Access Errors

Illegal access errors indicate that the wrong area was accessed in one of the following ways when the address specifying the instruction operand was accessed.

1,2,3...

1. A read or write was executed for a parameter area.
2. A write was executed in a memory area that is not mounted (See note.).
3. A write was executed in a read-only area.
4. The value specified in an indirect DM address in BCD mode was not BCD (e.g., *D1 contains #A000).

   Note An IR register containing the internal memory address of a bit is used as a word address or an IR containing the internal memory address of a word is used as a bit address.

Instruction processing will continue and the Error Flag (ER Flag) will not turn ON if an access error occurs, but the Access Error Flag (AER Flag) will turn ON.

If Instruction Errors are set to Stop Operation in the PLC Setup, then operation will stop (fatal error) and the “Illegal Access Error Flag” (A295.10) will turn ON if an illegal access error occurs and the AER Flag turns ON.

Note The Access Error Flag (AER Flag) will not be cleared after a task is executed. If Instruction Errors are set to Continue Operation, this Flag can be monitored until just before the END(001) instruction to see if an illegal access error has occurred in the task program. (The status of the final AER Flag after the entire user program has been executed will be monitored if the AER Flag is monitored on the CX-Programmer.)

Other Errors

Illegal Instruction Errors

Illegal instruction errors indicate that an attempt was made to execute instruction data other than that defined in the system. This error will normally not occur as long as the program is created on a the CX-Programmer.

In the rare even that this error does occur, it will be treated as a program error, operation will stop (fatal error), and the Illegal Instruction Flag (A295.14) will turn ON.

UM (User Memory) Overflow Errors

UM overflow errors indicate that an attempt was made to execute instruction data stored beyond the last address in the user memory (UM) defined as program storage area. This error will normally not occur as long as the program is created on the CX-Programmer.

In the rare even that this error does occur, it will be treated as a program error, operation will stop (fatal error), and the UM Overflow Flag (A295.15) will turn ON.

1-3-4 Checking Fatal Errors

The following errors are fatal program errors and the CPU Unit will stop running if one of these occurs. When operation is stopped by a program error, the task number where operation stopped will be stored in A294 and the program address will be stored in A298/A299. The cause of the program error can be determined from this information.

Address	Description	Stored Data
A294	The type of task and the task number at the point where operation stopped will be stored here if operation stops due to a program error. **Note** FFFF hex will be stored if there are no active cyclic tasks in a cycle, i.e., if there are no cyclic tasks to be executed.	Cyclic task: 0000 to 001F hex (cyclic tasks 0 to 31) Interrupt task: 8000 to 80FF hex (interrupt tasks 0 to 255)
A298/A299	The program address at the point where operation stopped will be stored here in binary if operation stops due to a program error. **Note** If the END(001) instruction is missing (A295.11 will be ON), the address where END(001) was expected will be stored. **Note** If there is a task execution error (A295.12 will be ON), FFFFFFFF hex will be stored in A298/A299.	A298: Rightmost portion of program address A299: Leftmost portion of program address

Note If the Error Flag or Access Error Flag turns ON, it will be treated as a program error and it can be used to stop the CPU from running. Specify operation for program errors in the PLC Setup.

Program error	Description	Related flags
No END Instruction	An END instruction is not present in the program.	The No END Flag (A295.11) turns ON.
Error During Task Execution	No task is ready in the cycle. No program is allocated to a task. The corresponding interrupt task number is not present even though the execution condition for the interrupt task was met.	The Task Error Flag (295.12) turns ON.
Instruction Processing Error (ER Flag ON) and Stop Operation set for Instruction Errors in PLC Setup	The wrong data values were provided in the operand when an attempt was made to execute an instruction.	The ER Flag turns ON and the Instruction Processing Error Flag (A295.08) turns ON if Stop Operation set for Instruction Errors in PLC Setup.
Illegal Access Error (AER Flag ON) and Stop Operation set for Instruction Errors in PLC Setup	A read or write was executed for a parameter area. A write was executed in a memory area that is not mounted. A write was executed in a read-only area. The value specified in an indirect DM address in BCD mode was not BCD.	AER Flag turns ON and the Illegal Access Error Flag (A295.10) turns ON if Stop Operation set for Instruction Errors in PLC Setup
Indirect DM BCD Error and Stop Operation set for Instruction Errors in PLC Setup	The value specified in an indirect DM address in BCD mode is not BCD.	AER Flag turns ON and the DM Indirect BCD Error Flag (A295.09) turns ON if Stop Operation set for Instruction Errors in PLC Setup
Differentiation Address Overflow Error	During online editing, more than 131,072 differentiated instructions have been inserted or deleted.	The Differentiation Overflow Error Flag (A295.13) turns ON.

Program error	Description	Related flags
UM (User Memory) Overflow Error	An attempt was made to execute instruction data stored beyond the last address in user memory (UM) defined as program storage area.	The UM (User Memory) Overflow Flag (A295.5) turns ON.
Illegal Instruction Error	An attempt was made to execute an instruction that cannot be executed.	The Illegal Instruction Flag (A295.14) turns ON.

1-4 Introducing Function Blocks

Function blocks can be used with CP-series CPU Units. Refer to the *CX-Programmer Ver. 7.0 Operation Manual Function Blocks* (W447) for details on actually using function blocks.

1-4-1 Overview and Features

Function blocks conforming to the IEC 61131-3 standard can be used with CX-Programmer Ver. 5.0 and higher. These function blocks are supported by CS/CJ-series CPU Units with unit version 3.0 or later and by CP-series CPU Units. The following features are supported.

- User-defined processes can be converted to block format by using function blocks.
- Function block algorithms can be written in the ladder programming language or in the structured text (ST) language. (See note.)
  - When ladder programming is used, ladder programs created with non-CX-Programmer Ver. 4.0 or earlier can be reused by copying and pasting.
  - When ST language is used, it is easy to program mathematical processes that would be difficult to enter with ladder programming.

  Note The ST language is an advanced language for industrial control (primarily Programmable Logic Controllers) that is described in IEC 61131-3. The ST language supported by CX-Programmer conforms to the IEC 61131-3 standard.
- Function blocks can be created easily because variables do not have to be declared in text. They are registered in variable tables.
  A variable can be registered automatically when it is entered in a ladder or ST program. Registered variables can also be entered in ladder programs after they have been registered in the variable table.
- A single function block can be converted to a library function as a single file, making it easy to reuse function blocks for standard processing.
- A program check can be performed on a single function block to easily confirm the function block's reliability as a library function.
- Programs containing function blocks (ladder programming language or structured text (ST) language) can be downloaded or uploaded in the same way as standard programs that do not contain function blocks. Tasks containing function blocks, however, cannot be downloaded in task units (uploading is possible).
- One-dimensional array variables are supported, so data handling is easier for many applications.

Note The IEC 61131 standard was defined by the International Electrotechnical Commission (IEC) as an international programmable logic controller (PLC) standard. The standard is divided into 7 parts. Specifications related to PLC programming are defined in *Part 3 Textual Languages (IEC 61131-3).*

- A function block (ladder programming language or structured text (ST) language) can be called from another function block (ladder programming language or structured text (ST) language). Function blocks can be nested up to 8 levels and ladder/ST language function blocks can be combined freely.

1-4-2 Function Block Specifications

For specifications that are not listed in the following table, refer to the *CX-Programmer Ver. 7.0 Operation Manual Function Blocks* (W447).

Item		Specifications
Model number		WS02-CXPC1-E-V6
Setup disk		CD-ROM
Compatible CPU Units		CP-series CPU Units with unit version 1.0 or later CS/CJ-series CS1-H, CJ1-H, and CJ1M CPU Units with unit version 3.0 or later are compatible. Device Type — CPU Type • CS1G-H — CS1G-CPU42H/43H/44H/45H • CS1H-H — CS1H-CPU63H/64H/65H/66H/67H • CJ1G-H — CJ1G-CPU42H/43H/44H/45H • CJ1H-H — CJ1H-CPU65H/66H/67H • CJ1M — CJ1M-CPU11/12/13/21/22/23
		CS/CJ/CP Series Function Restrictions • Instructions Not Supported in Function Block Definitions Block Program Instructions (BPRG and BEND), Subroutine Instructions (SBS, GSBS, RET, MCRO, and SBN), Jump Instructions (JMP, CJP, and CJPN), Step Ladder Instructions (STEP and SNXT), Immediate Refresh Instructions (!), I/O REFRESH (IORF), ONE-MS TIMER (TMHH).
Compatible computers	Computer	IBM PC/AT or compatible
	CPU	133 MHz Pentium or faster with Windows 98, 98SE, or NT 4.0 (with service pack 6 or higher)
	OS	Microsoft Windows 95, 98, 98SE, Me, 2000, XP, or NT 4.0 (with service pack 6 or higher)
	Memory	64 Mbytes min. with Windows 98, 98SE, or NT 4.0 (with service pack 6 or higher) Refer to the *CX-Programmer Ver. 7.0 Operation Manual* (W437) for details.
	Hard disk space	100 Mbytes min. available disk space
	Monitor	SVGA (800 × 600 pixels) min. **Note** Use “small font” for the font size.
	CD-ROM drive	One CD-ROM drive min.
	COM port	One RS-232C port min.

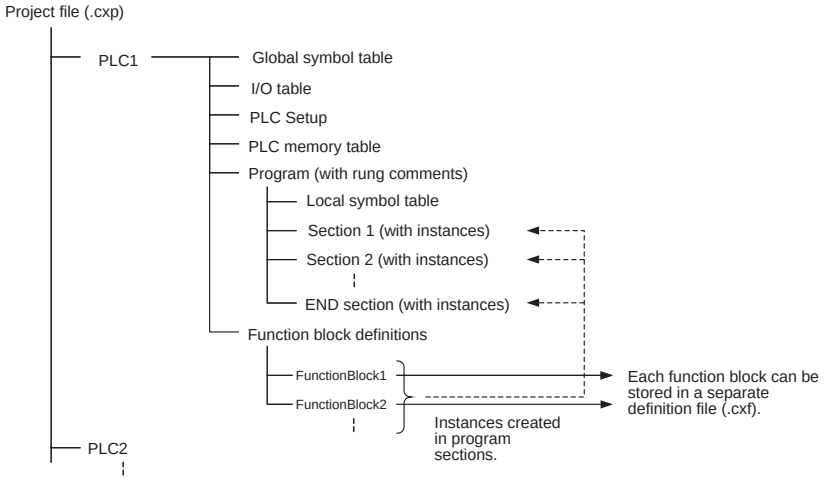
Note The structured text (ST language) conforms to the IEC 61131-3 standard, but CX-Programmer Ver. 5.0 supports only assignment statements, selection statements (CASE and IF statements), iteration statements (FOR, WHILE, REPEAT, and EXIT statements), RETURN statements, arithmetic operators, logical operators, comparison functions, numeric functions, and comments.

1-4-3 Files Created with CX-Programmer Ver. 7.0

Project Files (*.cxp) and File Memory Program Files (*.obj)

Projects created using CX-Programmer that contain function block definitions and projects with instances are saved in the same standard project files (*.cxp) and file memory program files (*.obj).

The following diagram shows the contents of a project. The function block definitions are created at the same directory level as the program within the relevant PLC directory.



Function Block Library Files (*.cxf)

A function block definition created in a project with CX-Programmer Ver. 7.0 can be saved as a file (1 definition = 1 file), enabling definitions to be loaded into other programs and reused.

Note When function blocks are nested, all of the nested (destination) function block definitions are included in this function block library file (.cxf).

Project Text Files Containing Function Blocks (*.cxt)

Data equivalent to that in project files created with CX-Programmer Ver. 7.0 (*.cxp) can be saved as CXT text files (*.cxt).

SECTION 2
Tasks

This section describes the operation of tasks and how to use tasks in programming.

2-1 Programming with Tasks

2-1-1 Overview

CP1H control operations can be divided by functions, controlled devices, processes, developers, or any other criteria and each operation can be programmed in a separate unit called a “task.” Using tasks provides the following advantages:

1,2,3...

1. Programs can be developed simultaneously by several people.

   Individually designed program parts can be assembled with very little effort into a single user program.

2. Programs can be standardized in modules.

   More specifically, the following the CX-Programmer functions will be combined to develop programs that are standalone standard modules rather than programs designed for specific systems (machines, devices). This means that programs developed separately by several people can be readily combine.

   - Programming using symbols
   - Global and local designation of symbols
   - Automatic allocation of local symbols to addresses

3. Improved overall response.

   Overall response is improved because the system is divided into an overall control program as well as individual control programs, and only specific programs will be executed as needed.

4. Easy revision and debugging.

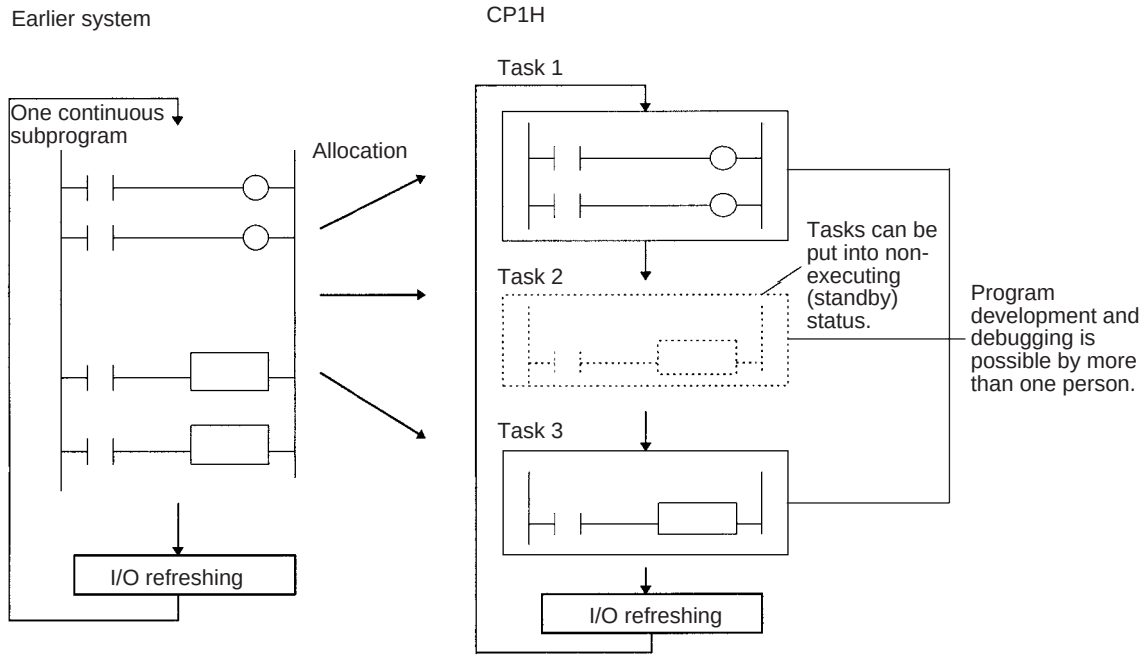
   - Debugging is much more efficient because tasks can be developed separately by several people, and then revised and debugged by individual task.
   - Maintenance is simple because only the task that needs revising will be changed in order to make specification or other changes.
   - Debugging is more efficient because it is easy to determine whether an address is specific or global and addresses between programs only need to be checked once during debugging because symbols are designated globally or locally and local symbols are allocated automatically to addresses through the CX-Programmer.

5. Easy to switch programs.

   A task control instruction in the program can be used to execute product-specific tasks (programs) when changing operation is necessary.

6. Easily understood user programs.

   Programs are structured in blocks that make the programs much simpler to understand for sections that would conventionally be handled with instructions like jump.



Note Unlike earlier programs that can be compared to reading a scroll, tasks can be compared to reading through a series of individual cards.

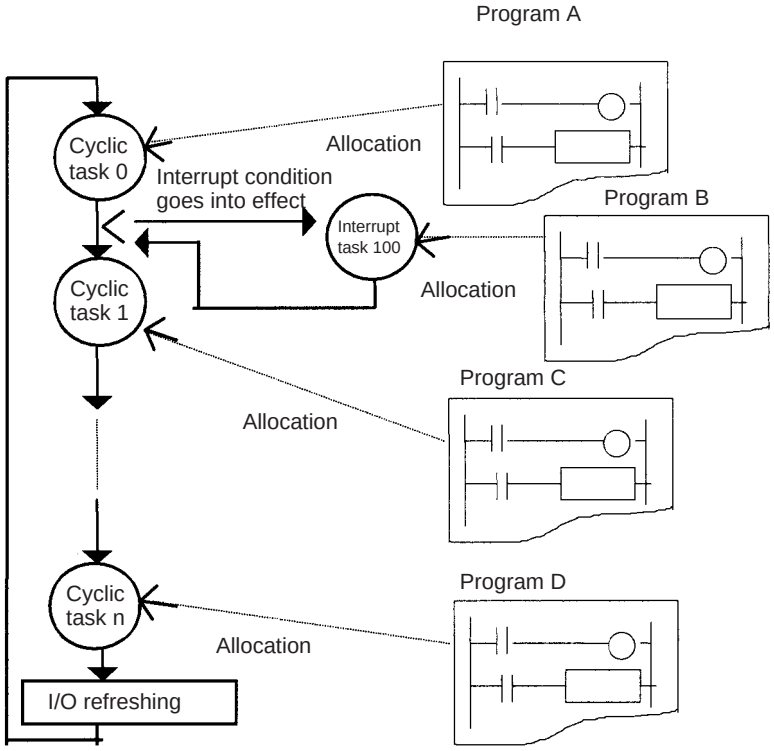
- All cards are read in a preset sequence starting from the lowest number.
- All cards are designated as either active or inactive, and cards that are inactive will be skipped. (Cards are activated or deactivated by task control instructions.)
- A card that is activated will remain activated and will be read in subsequent sequences. A card that is deactivated will remain deactivated and will be skipped until it is reactivated by another card.

2-1-2 Tasks and Programs

Up to 288 programs (tasks) can be controlled. Individual programs are allocated 1:1 to tasks. Tasks are broadly grouped into the following types:

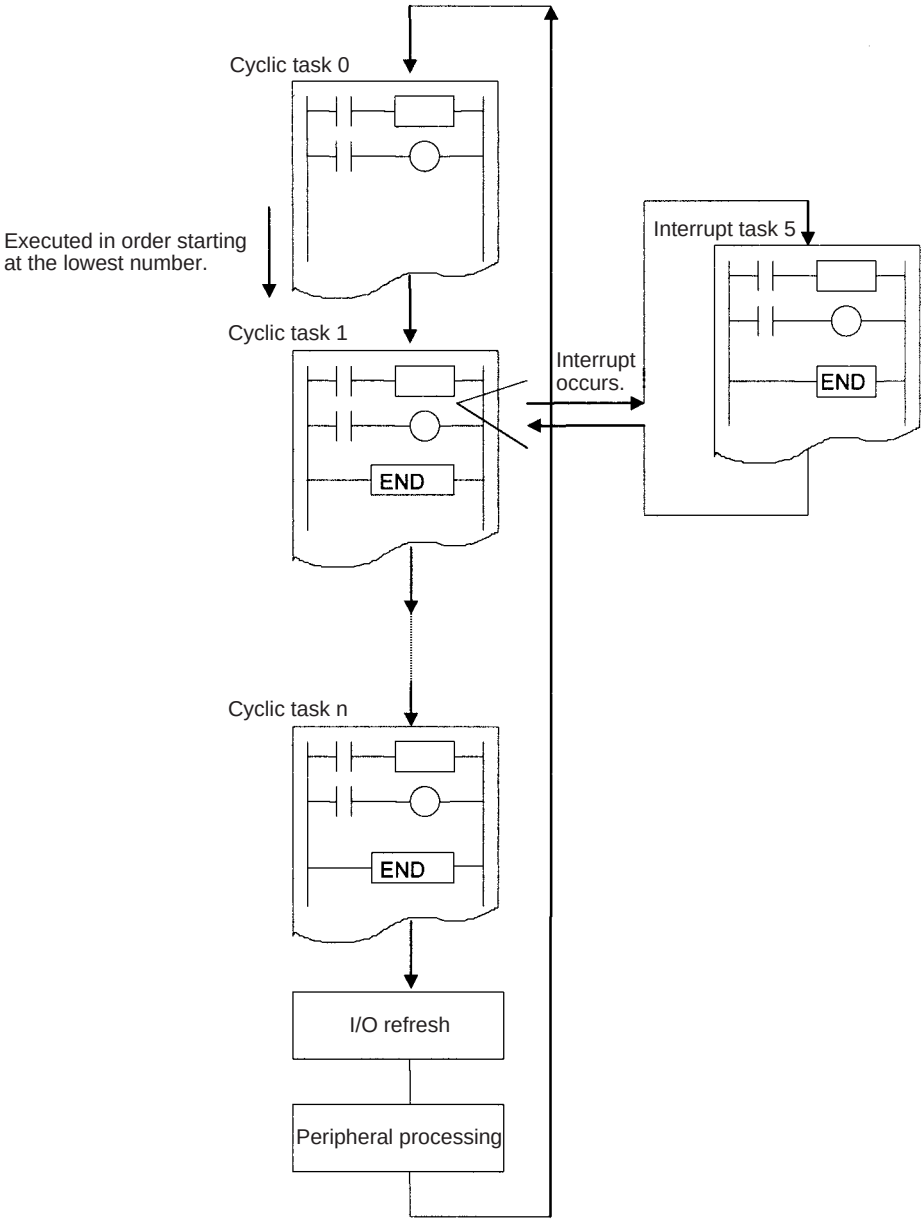
- Cyclic tasks
- Interrupt tasks

Each program allocated to a task is executed independently and must end with an END(001) instruction. I/O refreshing will be executed only after all task programs in a cycle have been executed.



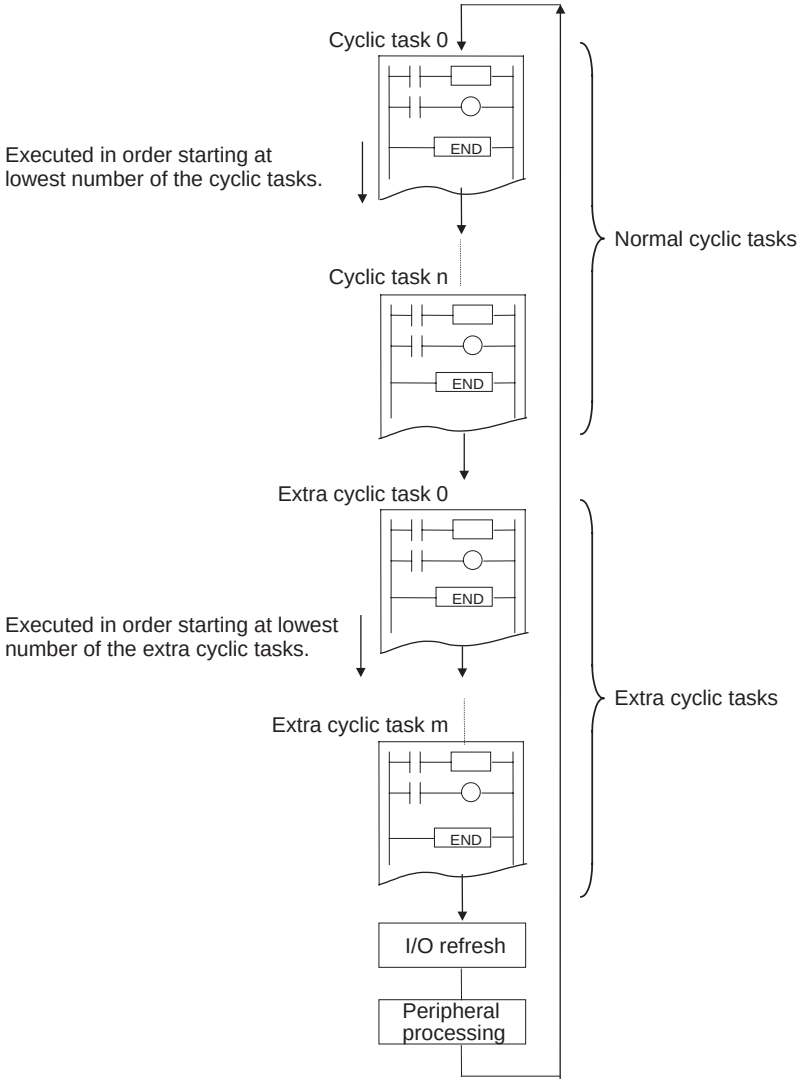
2-1-3 Basic CPU Unit Operation

The CPU Unit will execute cyclic tasks starting at the task with the lowest number. It will also interrupt cyclic task execution to execute an interrupt task if an interrupt occurs.



Note All Condition Flags (ER, CY, Equals, AER, etc.) and instruction conditions (interlock ON, etc.) will be cleared at the beginning of a task. Therefore Condition Flags cannot be read nor can INTERLOCK/INTERLOCK CLEAR (IL/ILC) instructions, JUMP/JUMP END (JMP/JME) instructions, or SUBROUTINE CALL/SUBROUTINE ENTRY (SBS/SBN) instructions be split between two tasks.

Interrupt task can be executed as cyclic tasks by starting them with TKON. These are called “extra cyclic tasks.” Extra cyclic tasks (interrupt task numbers 0 to 255) are executed starting at the lowest task number after execution of the normal cyclic task (celiac task numbers 0 to 31) has been completed.



2-1-4 Types of Tasks

Tasks are broadly classified as either cyclic tasks or interrupt tasks. Interrupt tasks are further divided into scheduled, input, high-speed counter, and external interrupt tasks. Interrupt tasks can also be executed as extra cyclic tasks.

Cyclic Tasks

A cyclic task that is READY will be executed once each cycle (from the top of the program until the END(001) instruction) in numerical order starting at the task with the lowest number. The maximum number of cyclic tasks is 32. (Cyclic task numbers: 00 to 31).

Interrupt Tasks

An interrupt task will be executed if an interrupt occurs even if a cyclic task (including extra cyclic tasks) is currently being executed. The interrupt task will be executed using any time in the cycle, including during user program execution, I/O refreshing, or peripheral servicing, when the execution condition for the interrupt is met.

Interrupt tasks can also be executed as extra cyclic tasks.

Input Interrupts (Direct Mode and Counter Mode)

An interrupt task can be executed each time one of the built-in inputs on the CPU Unit turns ON or OFF (Direct Mode) or when a specified number of inputs has been counted (Count Mode). For an X or XA CPU Unit, up to eight input interrupt tasks can be used (interrupt task numbers 140 to 147). For a Y CPU Unit, up to six input interrupt tasks can be used (interrupt task numbers 140, 141, or 144 to 147).

Scheduled Interrupt Tasks

A scheduled interrupt task will be executed at a fixed interval based on the internal timer of the CPU Unit. Only one scheduled interrupt tasks can be used (interrupt task number:2).

High-speed Counter Interrupts

Pulse inputs to a built-in high-speed counter in the CPU Unit can be counted to trigger execution of an interrupt.

External Interrupt Tasks

A user-specified interrupt task (interrupt task numbers 0 to 255) can be executed when an external interrupt occurs.

The interrupt task will be executed when requested by a user program running in a CJ-series Special I/O Unit or CJ-series CPU Bus Unit.

Up to 256 external interrupt tasks can be used (interrupt task numbers: 0 to 255). If an external interrupt task has the same number as scheduled, input, or high-speed counter interrupt task, the interrupt task will be executed for either condition (the two conditions will operate with OR logic) but basically task numbers should not be duplicated.

Note If another interrupt task is being executed when an input, scheduled, high-speed counter, or external interrupt occurs, then these interrupt tasks will not be executed until the interrupt task that is currently being executed has been completed. If multiple interrupts occur simultaneously, then interrupt tasks will be executed sequentially starting at the lowest interrupt task number.

Extra Cyclic Tasks

An interrupt tasks can be executed every cycle, just like the normal cyclic tasks. Extra cyclic tasks (interrupt task numbers 0 to 255) are executed starting at the lowest task number after execution of the normal cyclic task (cyclic task numbers 0 to 31) has been completed. The maximum number of extra cyclic tasks is 256 (Interrupt task numbers: 0 to 255). Cycle interrupt tasks, however, are different from normal cyclic tasks in that they are started with TKON(820), i.e., they cannot be started automatically at startup.

If an extra cyclic task has the same number as a scheduled, input, or high-speed counter interrupt task, the interrupt task will be executed for either condition (the two conditions will operate with OR logic). Do not use interrupt tasks both as normal interrupt tasks and as extra cyclic tasks.

Note

(1) Also, TKON(820) and TKOF(821) cannot be used in extra cyclic tasks, meaning that normal cyclic tasks and other extra cyclic tasks cannot be controlled from within an extra cyclic task.

(2) The differences between normal cyclic tasks and extra cyclic tasks are listed in the following table.

Item	Extra cyclic tasks	Normal cyclic tasks
Activating at startup	Not supported.	Supported. (Set from CX-Programmer.)
Using TKON(820) and TKOF(821) inside task	Not supported.	Supported.
Task Flags	Not supported.	Supported. (Cyclic task numbers 00 to 31 correspond to Task Flags TK00 to TK31.)

Item	Extra cyclic tasks	Normal cyclic tasks
Initial Task Execution Flag (A200.15) and Task Start Flag (A200.14)	Not supported.	Supported.
Index (IR) and data (DR) register values	Not defined when task is started (same as normal interrupt tasks). Values at the beginning of each cycle are undefined. Always set values before using them. Values set in the previous cycle cannot be read.	Undefined at the beginning of operation. Values set in the previous cycle can be read.

2-1-5 Task Execution Conditions and Settings

The following table describes task execution conditions, related settings, and status.

Task		No.	Execution condition	Related Setting
Cyclic tasks		0 to 31	Executed once each cycle if READY (set to start initially or started with the TKON(820)instruction) when the right to execute is obtained.	None
Interrupt tasks	Scheduled interrupt task 0	Interrupt task 2	Executed once every time the preset period elapses according to the internal timer of CPU Unit.	• The scheduled interrupt time is set (0 to 9999) through the SET INTERRUPT MASK instruction (MSKS(690)). • Scheduled interrupt unit (10 ms, 1.0 ms, or 0.1 ms) is set in PLC Setup.
	Input interrupt tasks 0 to 7	Interrupt tasks 140 to 147	Executed when the corresponding CPU Unit built-in input turns ON.	• Masks for designated inputs are canceled through the SET INTERRUPT MASK instruction (MSKS(690)).
	High-speed counter interrupt tasks	Interrupt tasks 0 to 255	Executed when corresponding target or range comparison condition is met for CPU Unit built-in high-speed counter.	
	External interrupt tasks	Interrupt tasks 0 to 255	Executed when requested by a user program in a Special I/O Unit or CPU Bus Unit.	None (always enabled)
Extra cyclic tasks 0 to 255		Interrupt tasks 0 to 255	Executed once each cycle if READY (started with the TKON(820) instruction) when the right to execute is obtained.	None (always enabled)

2-1-6 Cyclic Task Status

This section describes cyclic task status, including extra cyclic tasks.

Cyclic tasks always have one of four statuses: Disabled, READY, RUN (executable), and standby (WAIT).

Disabled Status (INI)

A task with Disabled status is not executed. All cyclic tasks have Disabled status in PROGRAM mode. Any cycle task that shifted from this to another status cannot return to this status without returning to PROGRAM mode.

READY Status

A task attribute can be set to control when the task will go to READY status. The attribute can be set to either activate the task using the TASK ON instruction or when RUN operation is started.

Instruction-activated Tasks

A TASK ON instruction (TKON(820)) is used to switch an instruction-activated cyclic task from Disabled status or Standby status to READY status.

Operation-activated Tasks

An operation-activated cyclic task will switch from Disabled status to READY status when the operating mode is changed from PROGRAM to RUN or MONITOR mode. This applies only to normal cyclic tasks.

Note The CX-Programmer can be used to set one or more tasks to go to READY status when operation is started for task numbers 0 through 31. The setting, however, is not possible with extra cyclic tasks.

RUN Status

A cyclic task that is READY will switch to RUN status and be executed when the task obtains the right to execute.

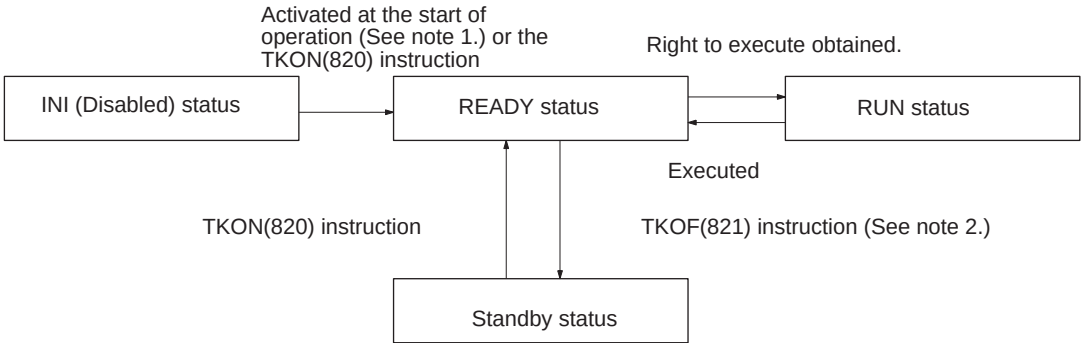
Standby Status

A TASK OFF (TKOF(821)) instruction can be used to change a cyclic task from Disabled status to Standby status.

Note The task programs for CP-series PLCs can be monitored online from the CX-Programmer to see if they are executing or stopped. The status indications on the CX-Programmer are as follows:

- Running: The task is in READY or RUN status. (There is no way to tell the difference between these.)
- Stopped: The task is in INI or WAIT status. (There is no way to tell the difference between these.)

2-1-7 Status Transitions

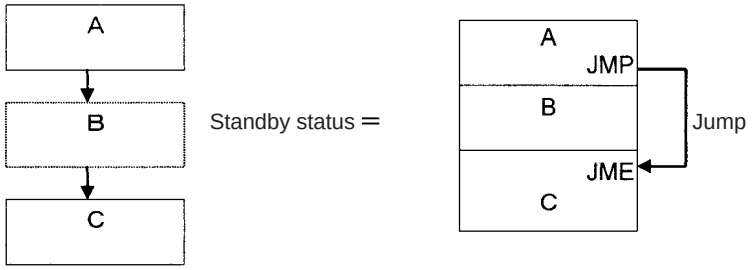


Note

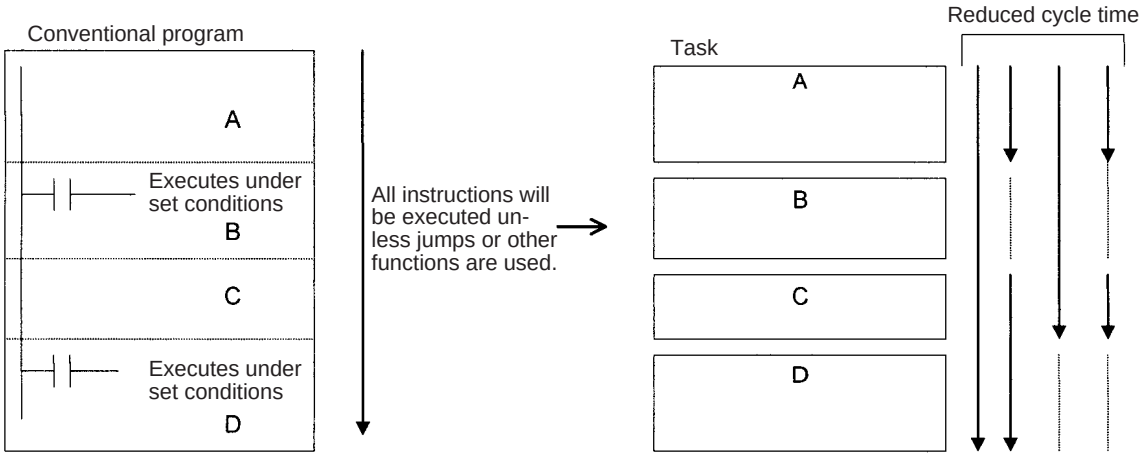
(1) Activation at the start of operation is possible for normal cyclic tasks only. It is not possible for extra cyclic tasks.

(2) A task in RUN status will be put into Standby status by the TKOF(821) instruction even when the TKOF(821) instruction is executed within that task.

Standby status functions exactly the same way as a jump (JMP-JME). Output status for the Standby task will be maintained.



Instructions will not be executed in Standby status, so instruction execution time will not be increased. Programming that does not need to be executed all the time can be made into tasks and assigned Standby status to reduce cycle time.



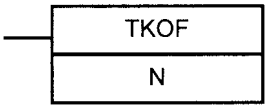
Note Standby status simply means that a task will be skipped during task execution. Changing to Standby status will not end the program.

2-2 Using Tasks

2-2-1 TASK ON and TASK OFF

The TASK ON (TKON(820)) and TASK OFF (TKOF(821)) instructions switch a cyclic task (including extra cyclic tasks) between READY and Standby status from a program.

N: Task No. A task will go to READY status when the execution condition is ON, and the corresponding Task Flag will turn ON.

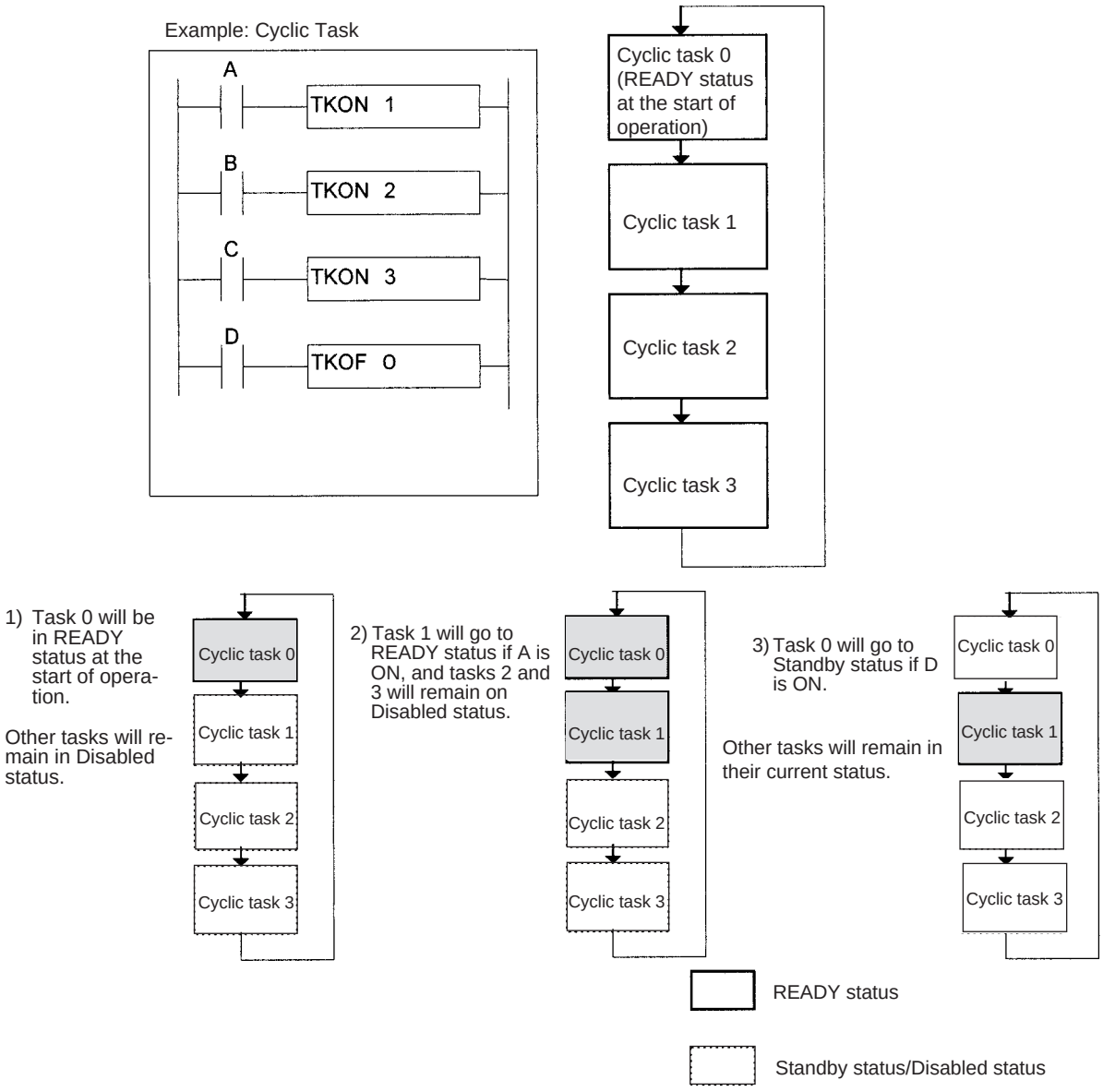


N: Task No. A task will go to Standby status when the execution condition is ON, and the corresponding Task Flag will turn OFF.

The TASK ON and TASK OFF instructions can be used to change any cyclic task between READY or Standby status at any time. A cyclic task that is in READY status will maintain that status in subsequent cycles, and a cyclic task that is in Standby status will maintain that status in subsequent cycles.

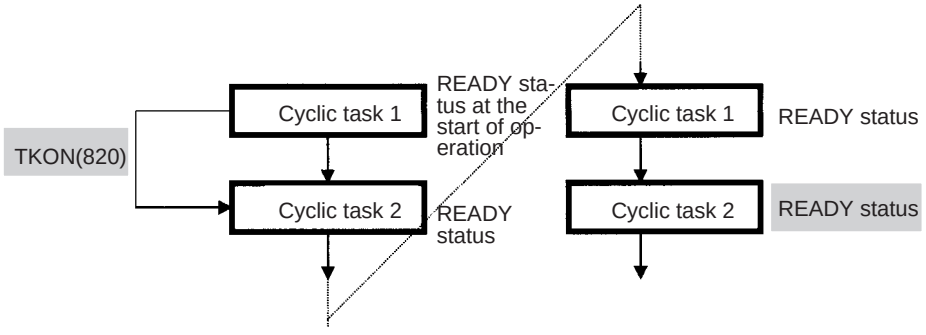
The TASK ON and TASK OFF instructions can be used only with cyclic tasks and not with interrupt tasks.

Note At least one cyclic task must be in READY status in each cycle. If there is not cyclic task in READY status, the Task Error Flag (A295.12) will turn ON, and the CPU Unit will stop running.

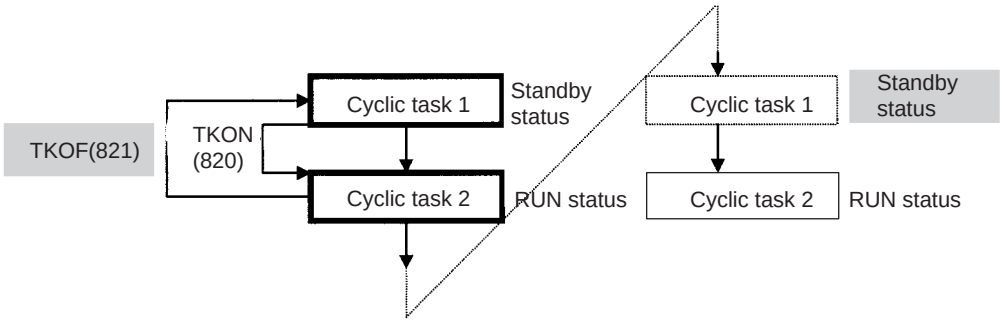


Tasks and the Execution Cycle

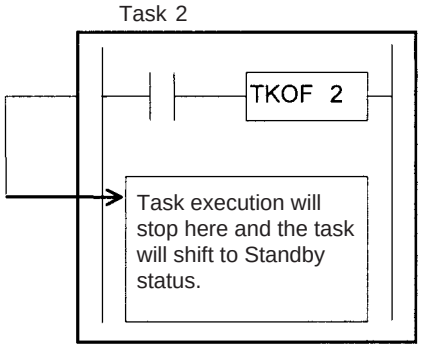
A cyclic task (including an extra cyclic task) that is in READY status will maintain that status in subsequent cycles.



A cyclic task that is in Standby status will maintain that status in subsequent cycles. The task will have to be activated using the TKON(820) instruction in order to switch from Standby to READY status.



If a TKOF(821) instruction is executed for the task it is in, the task will stop being executed where the instruction is executed, and the task will shift to Standby status.



Cyclic Task Numbers and the Execution Cycle (Including Extra Cyclic Tasks)

If task m turns ON task n and m > n, task n will go to READY status the next cycle.

Example: If task 5 turns ON task 2, task 2 will go to READY status the next cycle.

If task m turns ON task n and m < n, task n will go to READY status the same cycle.

Example: If task 2 turns ON task 5, task 5 will go to READY status in the same cycle.

If task m places task n in Standby status and m > n, will go to Standby status the next cycle.

Example: If task 5 places task 2 in Standby status, task 2 will go to Standby status the next cycle.

If task m places task n in Standby status and m < n, task n will go to Standby status in the same cycle.

Example: If task 2 places task 5 in Standby status, task 5 will go to Standby status in the same cycle.

Relationship of Tasks to I/O Memory

There are two different ways to use Index Registers (IR) and Data Registers (DR): 1) Independently by task or 2) Shared by all task.

With independent registers, IR0 used by cyclic task 1 for example is different from IR0 used by cyclic task 2. With shared registers, IR0 used by cyclic task 1 for example is the same as IR0 used by cyclic task 2.

The setting that determines if registers are independent or shared is made from the CX-Programmer.

- Other words and bits in I/O Memory are shared by all tasks. CIO 10.00 for example is the same bit for both cyclic task 1 and cyclic task 2. Therefore, be very careful in programming any time I/O memory areas other than the IR and DR Areas are used because values changed with one task will be used by other tasks.

I/O memory	Relationship to tasks
CIO, Auxiliary, Data Memory and all other memory areas except the IR and DR Areas.	Shared with other tasks.
Index registers (IR) and data registers (DR) (See note.)	Used separately for each task.

Note IR and DR values are not set when interrupt tasks (including extra cyclic tasks) are started. If IR and DR are used in an interrupt task, these values must be set by the MOVR/MOVRW (MOVE TO REGISTER and MOVE TIMER/COUNTER PV TO REGISTER) instructions within the interrupt task. After the interrupt task has been executed, IR and DR will return to their values prior to the interrupt automatically.

Relationship of Tasks to Timer Operation

Timer present values for TIM, TIMX, TIMH, TIMHX, TMHH, TMHHX, TIMW, TIMWX, TMHW, and TMHWX programmed for timer numbers T0000 to T2047 will be updated even if the task is switched or if the task containing the timer is changed to Standby status or back to READY status.

If the task containing TIM goes to Standby status and is the returned to READY status, the Completion Flag will be turned ON if the TIM instruction is executed when the present value is 0. (Completion Flags for timers are updated only when the instruction is executed.) If the TIM instruction is executed when the present value is not yet 0, the present value will continue to be updated just as it was while the task was in READY status.

- The present values for timers programmed with timer numbers T2048 to T4095 will be maintained when the task is in Standby status.

Relationship of Tasks to Condition Flags

All Condition Flags will be cleared before execution of each task. Therefore Condition Flag status at the end of task 1 cannot be read in task 2. CCS(282) and CCL(283) can be used to read Condition Flag status from another part of the program, e.g., from another task.

2-2-2 Task Instruction Limitations

Instructions Required in the Same Task

The following instructions must be placed within the same task. Any attempt to split instructions between two tasks will cause the ER Flag to turn ON and the instructions will not be executed.

Mnemonic	Instruction
JMP/JME	JUMP/JUMP END
CJP/JME	CONDITIONAL JUMP/JUMP END
CJPN/JME	CONDITIONAL JUMP NOT/CONDITIONAL JUMP END
JMP0/JME0	MULTIPLE JUMP/JUMP END
FOR/NEXT	FOR/NEXT
IL/ILC	INTERLOCK/INTERLOCK CLEAR
SBS/SBN/RET	SUBROUTINE CALL/SUBROUTINE ENTRY/SUBROUTINE RETURN
MCRO/SBN/RET	MACRO/SUBROUTINE ENTRY/SUBROUTINE RETURN
BPRG/BEND	BLOCK PROGRAM BEGIN/BLOCK PROGRAM END
STEP S/STEP	STEP DEFINE

Instructions Not Allowed in Interrupt Tasks

The following instructions cannot be placed in interrupt tasks. Any attempt to execute one of these instructions in an interrupt task will cause the ER Flag to turn ON and the instruction will not be executed.The following instructions can be used if an interrupt task is being used as an extra task.

Mnemonic	Instruction
TKON(820)	TASK ON
TKOF(821)	TASK OFF
STEP	STEP DEFINE
SNXT	STEP NEXT
STUP	CHANGE SERIAL PORT SETUP
DI	DISABLE INTERRUPT
EI	ENABLE INTERRUPT

The operation of the following instructions is unpredictable in an interrupt task: TIMER: TIM and TIMX(550), HIGH-SPEED TIMER: TIMH(015) and TIMHX(551), ONE-MS TIMER: TMHH(540) and TMHHX(552), ACCUMULATIVE TIMER: TTIM(087) and TTIMX(555), MULTIPLE OUTPUT TIMER: MTIM(543) and MTIMX(554), LONG TIMER: TIML(542) and TIMLX(553), TIMER WAIT: TIMW(813) and TIMWX(816), HIGH-SPEED TIMER WAIT: TMHW(815) and TMHWX(817), PID CONTROL: PID(190), FAILURE POINT DETECTION: FPD(269), and CHANGE SERIAL PORT SETUP: STUP(237).

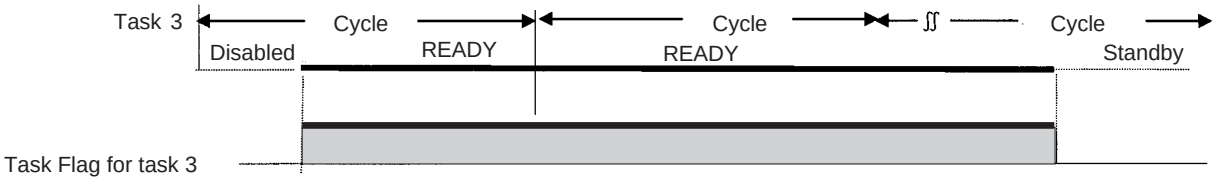
2-2-3 Flags Related to Tasks

Flags Related to Cyclic Tasks

The following flag work only for normal cyclic tasks. They do not work for extra cyclic tasks.

Task Flags (TK00 to TK31)

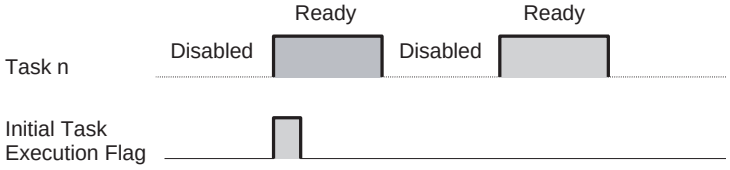
A Task Flag is turned ON when a cyclic task in READY status and is turned OFF when the task is in Disabled (INI) or in Standby (WAIT) status. Task numbers 00 to 31 correspond to Task Flags TK00 to TK31.



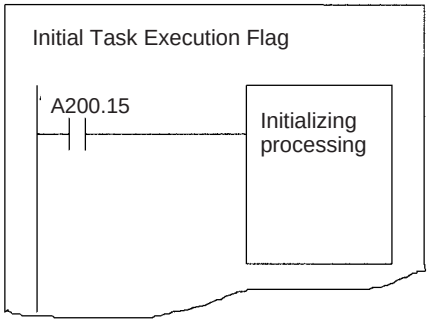
Note Task Flags are used only with cyclic tasks and not with interrupt tasks. With an interrupt task, A441.15 will turn ON if an interrupt task executes after the start of operation, and the number of the interrupt task that required for maximum processing time will be stored in two-digit hexadecimal in A441.00 to A441.07.

Initial Task Execution Flag (A200.15)

The Initial Task Execution Flag will turn ON when cyclic tasks shift from Disabled (INI) to READY status, the tasks obtain the right to execute, and the tasks are executed the first time. It will turn OFF when the first execution of the tasks has been completed.

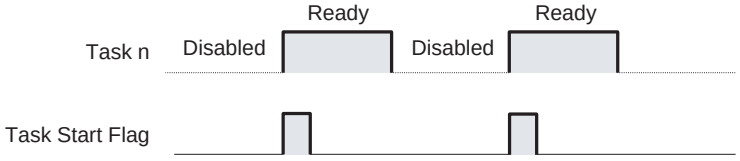


The Initial Task Execution Flag tells whether or not the cyclic tasks are being executed for the first time. This flag can thus be used to perform initialization processing within the tasks.

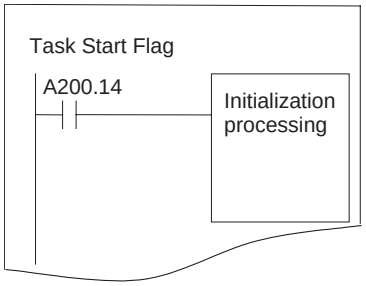


Note Even though a Standby cyclic task is shifted back to READY status through the TKON(820) instruction, this is not considered an initial execution and the Initial Task Execution Flag (A200.15) will not turn ON. The Initial Task Execution Flag (A200.15) will also not turn ON if a cyclic task is shifted from Disabled to RUN status or if it is put in Standby status by another task through the TKOF(821) instruction before the right to execute actually is obtained.

Task Start Flag (A200.14) The Task Start Flag can be used to perform initialization processing each time the task cycle is started. The Task Start Flag turns OF whenever cycle task status changes from Disabled (INI) or Standby (WAIT) status to READY status (whereas the Initial Task Execution Flag turns ON only when status changes from Disabled (INI) to READY).



The Task Start Flag can be used to perform initialization processing whenever a task goes from Standby to RUN status, i.e., when a task on Standby is enabled using the TRON(820) instruction.



Flags Related to All Tasks

Task Error Flag (A295.12) The Task Error Flag will turn ON if one of the following task errors occurs.

- No cyclic tasks (including extra cyclic tasks) are READY during a cycle.
- The program allocated to a cyclic task (including extra cyclic tasks) does not exist. (This situation will not occur when using the CX-Programmer.)
- No program is allocated to an activated interrupt task.

Task Number when Program Stopped (A294)

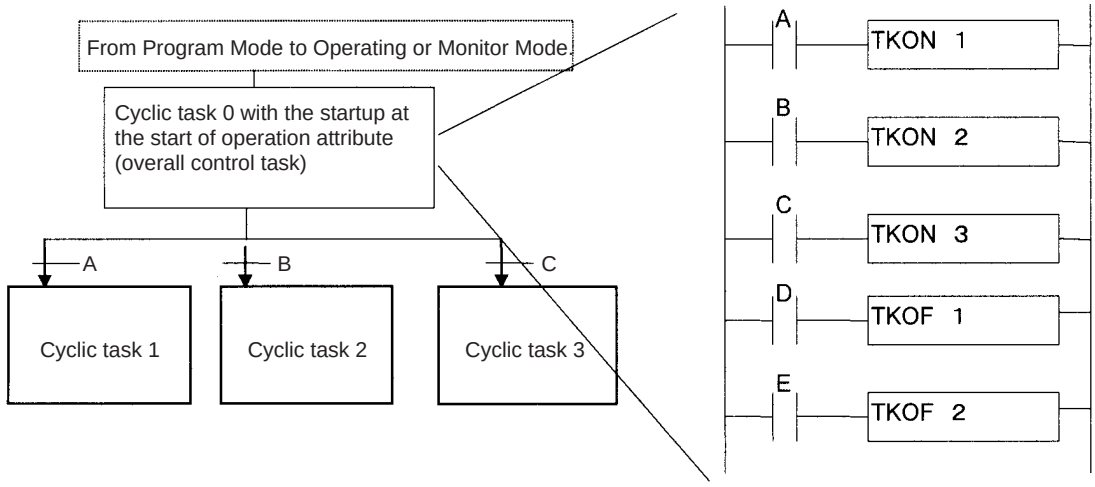
The type of task and the current task number when a task stops execution due to a program error will be stored as follows:

Type	A294
Cyclic task	0000 to 001F hex (correspond to task numbers 0 to 31)
Interrupt task	8000 to 80FF hex (correspond to interrupt task numbers 0 to 255)

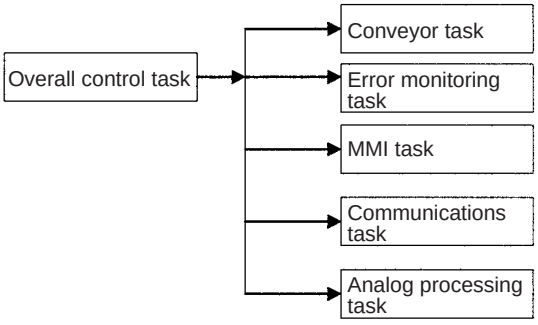
This information makes it easier to determine where the fatal error occurred, and it will be cleared when the fatal error is cleared. The program address where task operation stopped is stored in A298 (rightmost bits of the program address) and in A299 (leftmost bits of the program address).

2-2-4 Examples of Tasks

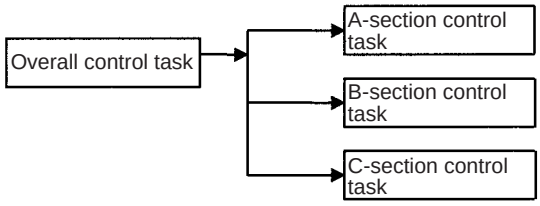
An overall control task that is set to go to READY status at the start of operation is generally used to control READY/Standby status for all other cyclic tasks (including extra cyclic tasks). Of course, any cyclic task can control the READY/Standby status of any other cyclic task as required by the application.



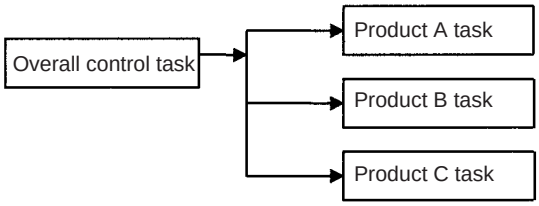
Tasks Separated by Function



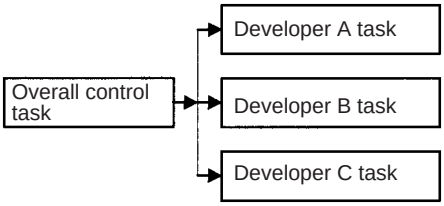
Tasks Separated by Controlled Section



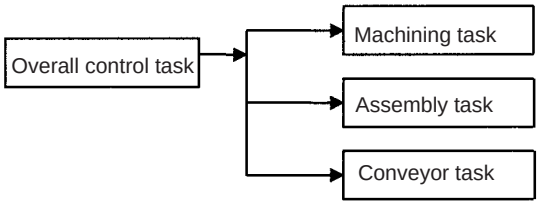
Tasks Separated by Product



Tasks Separated by Developer



Tasks Separated by Process



Combinations of the above classifications are also possible, e.g., classification by function and process.

2-2-5 Designing Tasks

We recommend the following guidelines for designing tasks.

1,2,3... 1. Use the following standards to study separating tasks.

a. Summarize specific conditions for execution and non-execution.

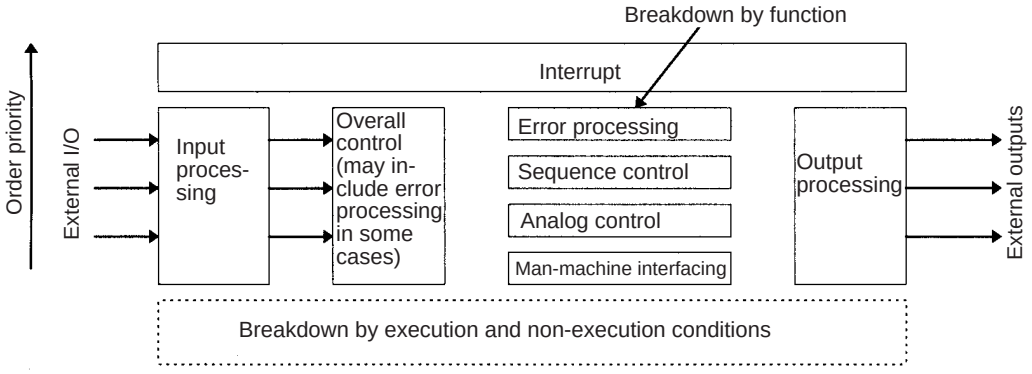
b. Summarize the presence or absence of external I/O.

c. Summarize functions.

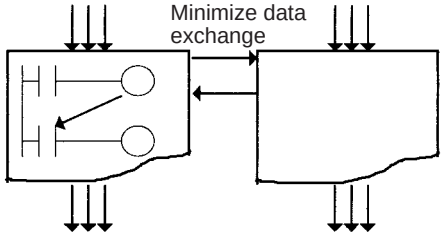
Keep data exchanged between tasks for sequence control, analog control, man-machine interfacing, error processing and other processes to an absolute minimum in order to maintain a high degree of autonomy.

d. Summarize execution in order of priority.

Separate processing into cyclic and interrupt tasks.



2. Be sure to break down and design programs in a manner that will ensure autonomy and keep the amount of data exchanged between tasks (programs) to an absolute minimum.



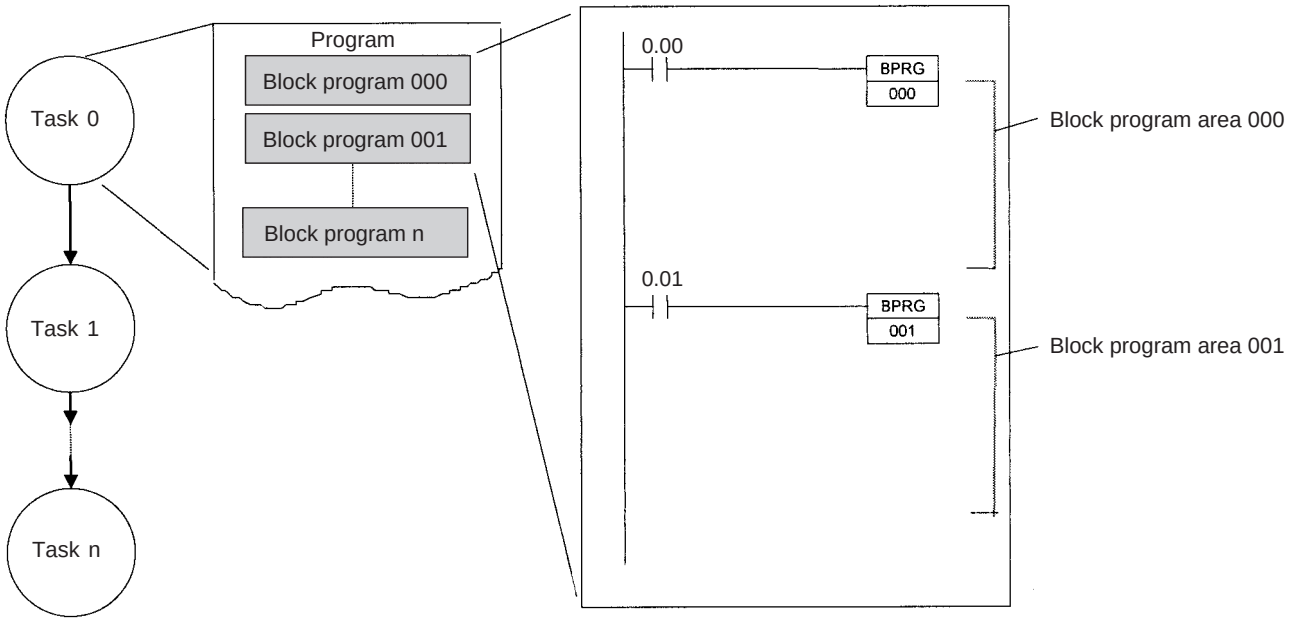
3. Generally, use an overall control task to control the READY/Standby status of the other tasks.
4. Allocate the lowest numbers to tasks with the highest priority.
Example: Allocate a lower number to the control task than to processing tasks.
5. Allocate lower numbers to high-priority interrupt tasks.
6. A task in READY status will be executed in subsequent cycles as long as the task itself or another task does not shift it to Standby status. Be sure to insert a TKOF(821) (TASK OFF) instruction for other tasks if processing is to be branched between tasks.
7. Use the Initial Task Execution Flag (A200.15) or the Task Start Flag (A200.14) in the execution condition to execution instructions to initialize tasks. The Initial Task Execution Flag will be ON during the first execution of each task. The Task Start Flag each time a task enters READY status.

8. Assign I/O memory into memory shared by tasks and memory used only for individual tasks, and then group I/O memory used only for individual tasks by task.

Relationship of Tasks to Block Programs

Up to 128 block programs can be created in the tasks. This is the total number for all tasks. The execution of each entire block program is controlled from the ladder diagram, but the instructions within the block program are written using mnemonics. In other words, a block program is formed from a combination of a ladder instruction and mnemonic code.

Using a block program makes it easier to write logic flow, such as conditional branching and process stepping, which can be hard to write using ladder diagrams. Block programs are located at the bottom of the program hierarchy, and the larger program units represented by the task can be split into small program units as block programs that operate with the same execution condition (ON condition).

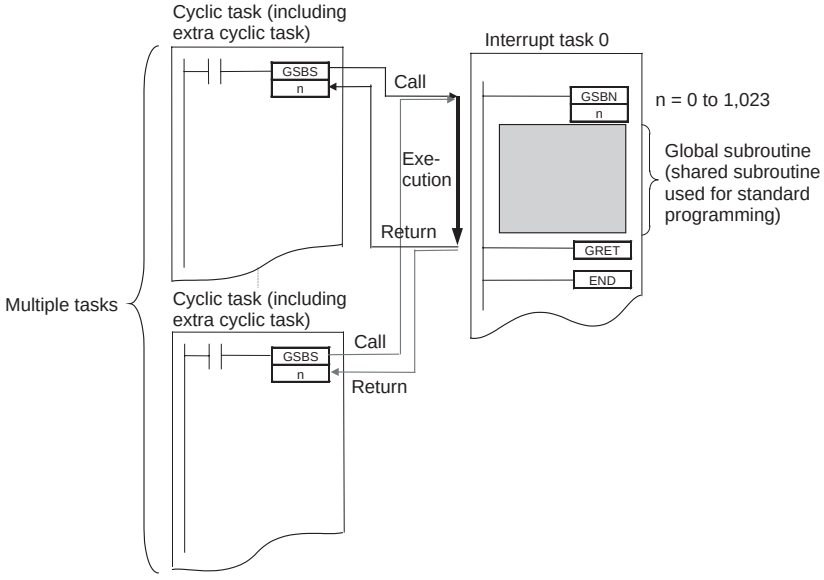


2-2-6 Global Subroutine

A subroutine in one task cannot be called from other tasks. A subroutine called a global subroutine can be created in interrupt task number 0, and this subroutine can be called from cyclic tasks (including extra cyclic tasks).

GSBS(750) is used to call a global subroutine. The subroutine number must be between 0 and 1,023. The global subroutine is defined at the end of interrupt task number 0 (just before END(001)) between GSBN(751) and GRET(752) instructions.

Global subroutines can be used to create a library of standard program sections that can be called whenever necessary.



2-3 Interrupt Tasks

2-3-1 Types of Interrupt Tasks

List of Interrupt Tasks

Type	Task No.	Execution condition	Setting procedure	Number of interrupts	Application examples
Input Interrupts 0 to 7	140 to 147	An interrupt occurs when an interrupt input built into the CPU Unit turns ON or OFF in Direct Mode or when a specified number of ON or OFF signals is detected for the interrupt input in Counter Mode.	Use the SET INTERRUPT MASK instruction MSKS(690) to enable interrupt inputs.	8 points	Increasing response speed to specific inputs
High-speed counter interrupts	0 to 255	An interrupt occurs when a condition is met for a target value or range comparison for the present value of a high-speed counter.	Use the COMPARISON TABLE LOADinstruction (CTBL(882)) to specify the execution condition and the interrupt to execute.	256 points	Performing positioning operations based on counting encoder pulses
Scheduled Interrupt 0	2	An interrupt occurs at a scheduled time (fixed intervals).	Use the SET INTERRUPT MASK instruction (MSKS(690)) to set the interrupt interval. See *Scheduled interrupt interval* in PLC Setup.	1 point	Monitoring operating status at fixed intervals
External Interrupts	0 to 255	Interrupts are requested by an Special I/O Unit or CPU Bus Unit.	None (always valid)	256 points	Performing processing required by CJ-series Special I/O Units

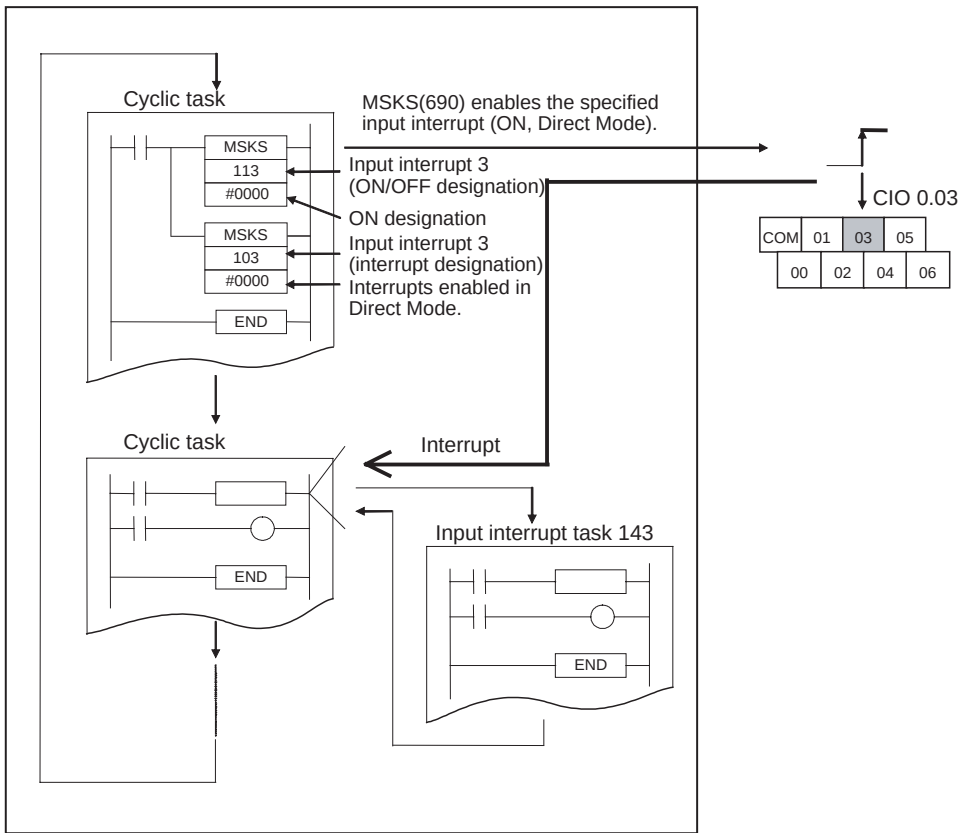
Input Interrupt Tasks: Tasks 140 to 147

Input interrupt tasks are disabled by default when cyclic task execution is started. To enable input interrupts, execute the SET INTERRUPT MASK instruction (MSKS(690)) in a cyclic task for the interrupt number.

Using inputs as interrupt inputs must be enabled in advance in the PLC Setup.

Note Do not enable unneeded input interrupt tasks. If the interrupt input is triggered by noise and there isn't a corresponding interrupt task, a fatal error (task error) will cause the program to stop.

Example: The following example shows execution input interrupt task 143 when CIO 0.03 (interrupt input No. 3) turns ON.



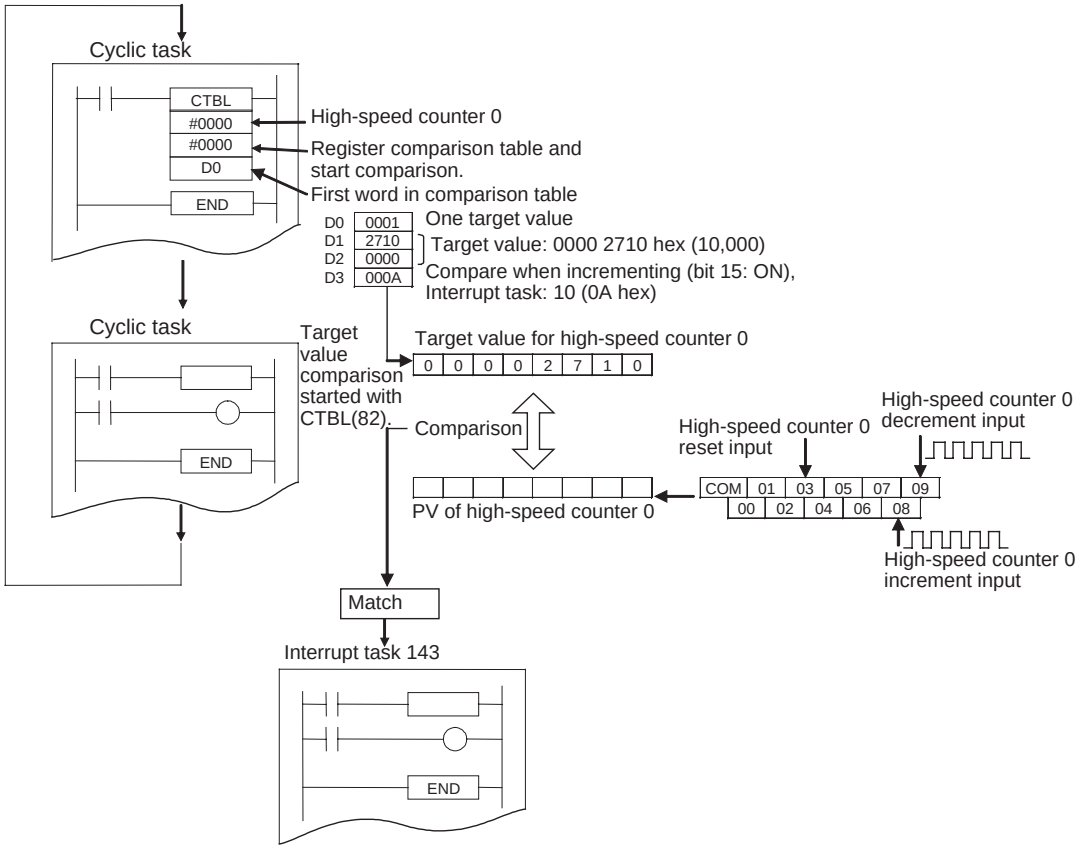
Interrupt input	Input interrupt number	Interrupt task number
CIO 0.00	0	140
CIO 0.01	1	141
CIO 0.02	2	142
CIO 0.03	3	143
CIO 1.00	4	144
CIO 1.01	5	145
CIO 1.02	6	146
CIO 1.03	7	147

High-speed Counter Interrupt Tasks: Tasks 0 to 255

High-speed counter interrupt tasks are enabled by executing the COMPARISON TABLE LOADinstruction (CTBL(882)) to specify the execution condition and the interrupt to execute. The comparison condition consists of target values or a comparison range.

Example

The following example illustrates executing high-speed interrupt task 10 when the present value of high-speed counter 0 equals the target value when the present value is incremented.



Scheduled Interrupt Task: Task 2

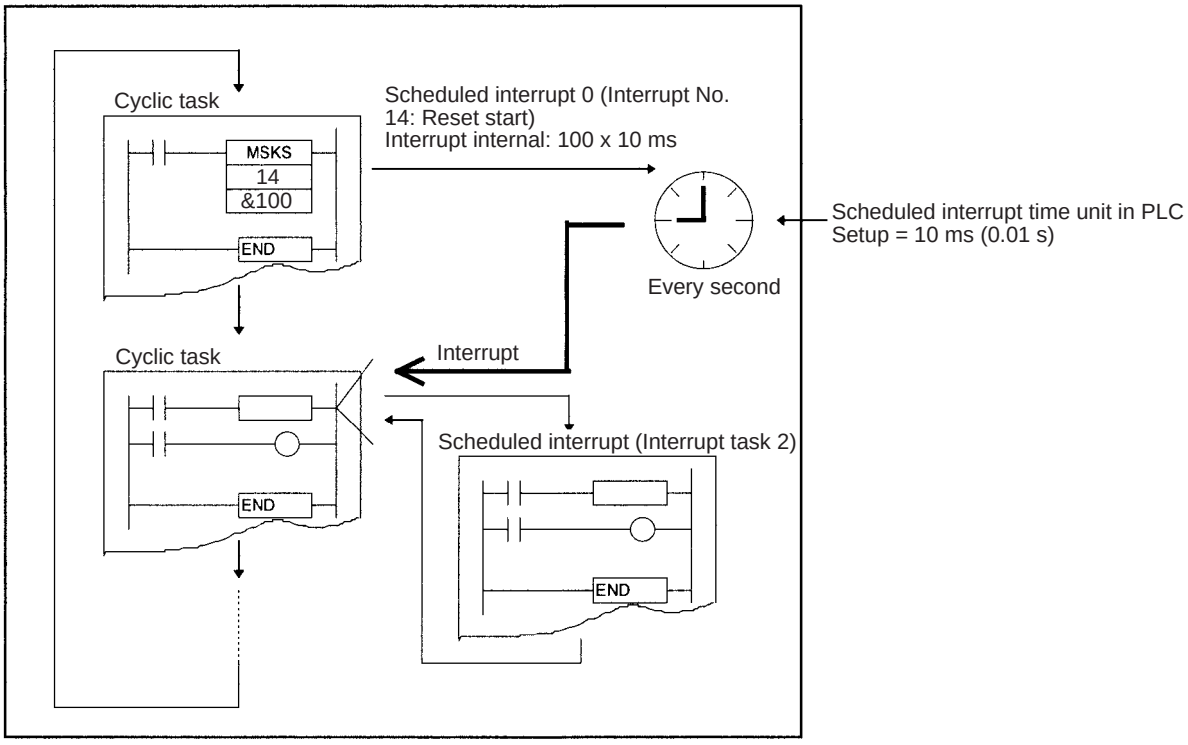
Scheduled interrupt tasks are disabled in the default PLC Setup at the start of cyclic task execution. Perform the following steps to enable scheduled interrupt tasks.

1,2,3...

1. Execute the SET INTERRUPT MASK instruction MSKS(690) from a cyclic task and set the time (cycle) for the specified scheduled interrupt.
2. Set the *Scheduled interrupt interval* in PLC Setup.

Note The interrupt time setting affects the cyclic task in that the shorter the interrupt time, the more frequently the task executes and the longer the cycle time.

Example: The following examples shows executed scheduled interrupt task every second.



Scheduled Interrupt Numbers and Interrupt Task Number

Scheduled interrupt number	interrupt task
0	2

PLC Setup Settings

Set the *Scheduled interrupt interval* on the Timings Tab Page of the PLC Setup to 0.1, 1.0, or 10 ms.

Name	Settings
Scheduled interrupt interval	10 ms (default)
	1.0 ms
	0.1 ms

External Interrupt Tasks: Tasks 0 to 255

External interrupt tasks can be received at any time. External interrupt processing is performed at the CPU Unit in PLCs containing CJ-series Special I/O Units or CPU Bus Units. Settings don't have to be made in the CPU Unit. The specified interrupt task must be programmed in the CPU Unit.

Note If an external interrupt task (0 to 255) has the same number as the scheduled interrupt task (task), an input interrupt task (140 to 147), or a high-speed counter task (0 to 255), the interrupt task will be executed for either interrupt condition (external interrupt or the other interrupt condition). As a rule, task numbers should not be duplicated.

Interrupt Task Priority and Order of Execution

All interrupt tasks have the same priority, i.e., once execution of any interrupt task has started, it will be completed through the end of the task even if another interrupt occurs during execution. For example, execution of an input interrupt task will not be interrupted to execute the scheduled interrupt task, i.e., the scheduled interrupt task will be executed only after completing the input interrupt task.

If more than one interrupt occurs at the same time, the interrupt tasks will be executed in the following order: Input interrupt tasks (Direct Mode or Counter Mode), High-speed interrupt tasks, External interrupt tasks, Scheduled interrupt task.

If more than one of the same type of interrupt occurs at the same time, the one with the lower task number will be executed first.

Keep in mind that the above order of execution means that time may be required to execute a programmed task even after an interrupt has occurred if the user program allows the possibility of more than one interrupt occurring at the same time. For example, the user must give special consideration to the scheduled interrupt, which may not be executed at the expected time if other interrupts occur.

2-3-2 Interrupt Task Flags and Words

Maximum Interrupt Task Processing Time (A440)

The maximum processing time for an interrupt task is stored in binary data in 0.1-ms units and is cleared at the start of operation.

Interrupt Task with Maximum Processing Time (A441)

The interrupt task number with maximum processing time is stored in binary data. Here, 8000 to 80FF hex correspond to task numbers 00 to FF hex.

A441.15 will turn ON when the first interrupt occurs after the start of operation. The maximum processing time for subsequent interrupt tasks will be stored in the rightmost two digits in hexadecimal and will be cleared at the start of operation.

Interrupt Task Error Flag (Nonfatal Error) (A402.13)

If Interrupt Task Error Detection is turned ON in the PLC Setup, the Interrupt Task Error Flag will turn ON if an interrupt task error occurs.

Interrupt Task Error Flag (A426.15)/Task Number Generating the Interrupt Task Error (A426.00 to A426.11)

If A402.13 turns ON, then the following data will be stored in A426.15 and A426.00 to A426.11.

A402.13	Interrupt Task Error Description	A426.15	A426.00 to A426.11
Interrupt Task Error (If Interrupt Task Error Detection is turned ON in the PLC Setup)	When trying to refresh I/O for a large number of words using IORF(097) from an interrupt task while a CJ-series Special I/O Unit is being refreshed by cyclic I/O refreshing.	ON	The unit number of the CJ-series Special I/O Unit being refreshed will be stored in 12 bits of binary data (unit number 0 to 95: 000 to 05F hex).

Task Number when Program Stopped (A294)

The type of task and the current task number when a program stops due to a program error will be stored in the following locations.

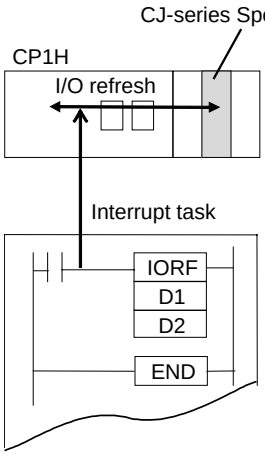
Type	A294
Interrupt task	8000 to 80FF hex (corresponds to interrupt task 0 to 255)
Cyclic task	0000 to 001F hex (corresponds to task 0 to 31)

2-3-3 Application Precautions

Executing IORF(097) for a Special I/O Unit

If an IORF(097) instruction has to be executed from an interrupt task for a CJ-series Special I/O Unit, be sure to turn OFF cyclic refresh for the Special I/O Unit (using the unit number) in the PLC Setup.

A interrupt task error will occur if you try to refresh a Special I/O Unit with an IORF(097) instruction from an interrupt task while the Unit is also being refreshed by cyclic I/O refresh or by I/O refresh instructions (IORF(097) or immediate refresh instructions (!)). If Interrupt Task Error Detection is turned ON in the PLC Setup when an interrupt task error occurs, A402.13 (Interrupt Task Error Flag) will turn ON and the unit number of the Special I/O Unit for which I/O refreshing has been duplicated will be stored in A426 (Interrupt Task Error, Task Number). The CPU Unit will continue running.



Incorrect Use

Do not executed IORF(097) in an interrupt task if cyclic refreshing is enabled for Special I/O Units in the PLC Setup.

Correct Use

Disable cyclic refreshing for Special I/O Units in the PLC Setup before executing the IORF(097) instruction in an interrupt task.

PLC Setup Settings

Select or clear the *Detect Interrupt Task Error* Checkbox in the *Execute Process* Area on the Settings Tab Page in the PLC Setup.

Name	Setting	Description
Detect Interrupt Task Error	Cleared	Interrupt task errors not detected.
	Selected	Interrupt Task Error Flag (A402.13) turned ON when an interrupt task error is detected.

Related Auxiliary Area Flags/Words

Name	Address		Description
Interrupt Task Error Flag	A402.13		Turns ON if you try to refresh a CJ-series Special I/O Unit with IORF(097) from an interrupt task while that Unit is being refreshed by cyclic I/O refresh.
Interrupt Task Error Unit Number	A426	Bits 00 to 11	The unit number of the Special I/O Unit undergoing duplicate refreshing will be stored here when A402.13 turns ON.
Interrupt Task Error Factor Flag		Bit 15	Turns ON to indicate the cause of the error when A402.13 turns ON.

Disabling Interrupts

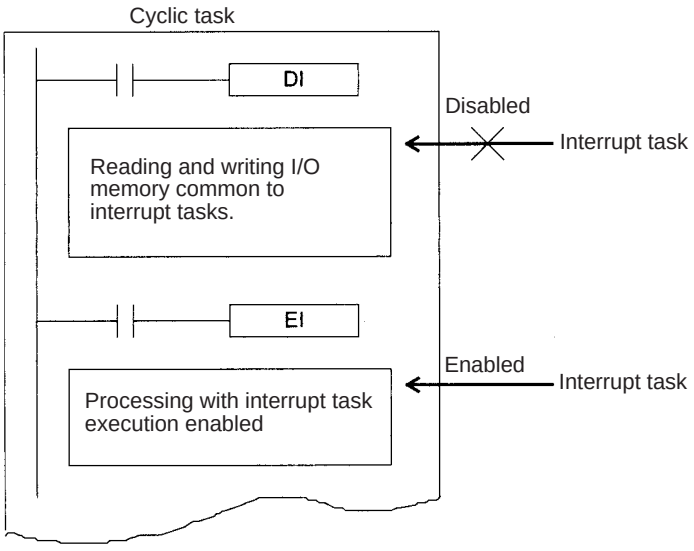
The following processing will be interrupted to execute an interrupt task.

- Instruction execution
- Refreshing for CPU Unit built-in I/O, CPM1A Expansion Units, CPM1A Expansion I/O Units, or CJ-series Special I/O Units
- Peripheral servicing

Data Concurrency between Cyclic and Interrupt Tasks

Data may not be concurrent if a cyclic (including extra cyclic tasks) and an interrupt task are reading and writing the same I/O memory addresses. Use the following procedure to disable interrupts during memory access by cyclic task instructions.

- Immediately prior to reading or writing by a cyclic task instruction, use a DISABLE INTERRUPT (DI(693)) instruction to disable execution of interrupt tasks.
- Use an ENABLE INTERRUPT instruction (EI(694)) immediately after processing in order to enable interrupt task execution.



Problems may occur with data concurrency even if DI(693) and EI(694) are used to disable interrupt tasks during execution of an instruction that requires response reception and processing (such as a network instruction or serial communications instruction).

2-4 CX-Programmer Operations for Tasks

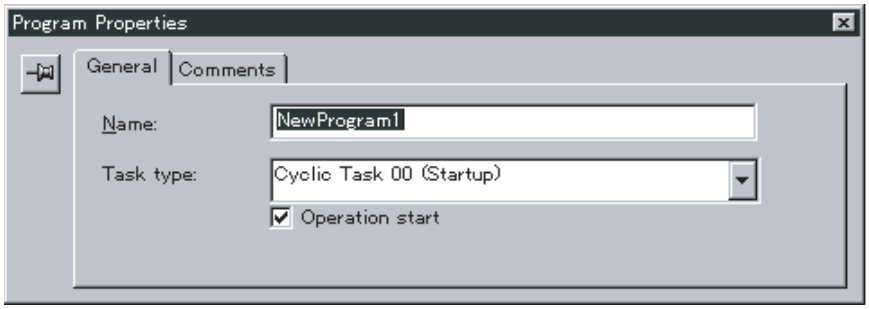
Note Use the CX-Programmer to create cyclic tasks (including extra cyclic tasks). Be sure to use the CX-Programmer to allocate the task types and task numbers.

CX-Programmer

Specify the task type and number as attributes for each program.

1,2,3...

1. Select ***View/Properties,*** or click the right button and select ***Properties*** from the popup menu, to display the program that will be allocated a task.
2. Click the **General** Tab, and select the *Task Type* and *Task No*. For a cyclic task, select the ***Operation start*** Option to start the task at startup if required.



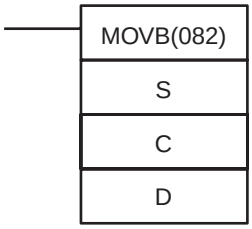
SECTION 3
Instructions

This section describes each of the instructions that can be used in programming CP-series PLCs. Instructions are described in order of function.

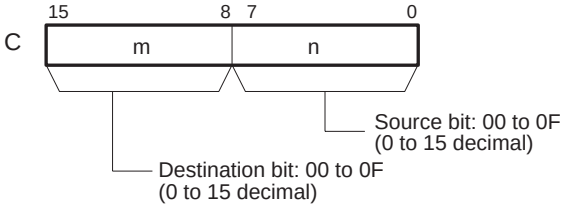
3-1 Notation and Layout of Instruction Descriptions

Instructions are described in groups by function. Refer to *Appendix C 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 colspan="2">Item</th><th>Contents</th></tr>
<tr><td colspan="2">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 colspan="2">Purpose</td><td>The basic purpose of the instruction is described after the section heading.</td></tr>
<tr><td colspan="2">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 rowspan="2">Variations</td><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 rowspan="3">Variations</td><td>Executed Each Cycle for ON Condition</td><td>MOVB(082)</td></tr><tr><td>Executed Once for Upward Differentiation</td><td>@MOVB(082)</td></tr><tr><td>Executed Once for Downward Differentiation</td><td>Not supported</td></tr></table></td></tr>
<tr><td>Variations</td><td>Immediate Refreshing Specification</td><td>Immediate refreshing can be specified for some instructions to refresh I/O when the instruction is executed. If immediate refreshing is supported, the specification is given using the mnemonic form. If immediate refreshing is not support by an instruction “Not supported” is given.
<table><tr><td>Immediate Refreshing Specification</td><td>Not supported.</td></tr></table></td></tr>
<tr><td colspan="2">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 0 to CIO 6143</td></tr>
<tr><td>Work Area</td><td colspan="3">W0 to W511</td></tr>
<tr><td>Holding Bit Area</td><td colspan="3">H0 to H511</td></tr>
<tr><td>Auxiliary Bit Area</td><td colspan="2">A0 to A959</td><td>A448 to A959</td></tr>
<tr><td>Timer Area</td><td colspan="3">T0000 to T4095</td></tr>
<tr><td>Counter Area</td><td colspan="3">C0000 to C4095</td></tr>
<tr><td>DM Area</td><td colspan="3">D0 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.

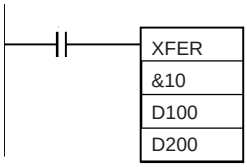
Operand Descriptions and Operand Specifications

- 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 1023” 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 for with an & prefix, as shown in the following example.



The input methods for constants for 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

Flag names are used for condition flags in this section. With the CX-Programmer, the condition flags are registered in advance as global symbols.

Flag name (Used in this section.)	CX-Programmer label
Error Flag	P_ER
Access Error Flag	P_AER
Carry Flag	P_CY
Greater Than Flag	P_GT
Equals Flag	P_EQ
Less Than Flag	P_LT
Negative Flag	P_N
Overflow Flag	P_OF
Underflow Flag	P_UF
Greater Than or Equals Flag	P_GE
Not Equal Flag	P_NE
Less Than or Equals Flag	P_LE
Always ON Flag	P_On
Always OFF Flag	P_Off

3-2 Sequence Input Instructions

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
Immediate Refreshing Specification		!LD
Combined Variations	**Refreshes Input Bit, Restarts Logic, and Creates ON Once for Upward Differentiation**	!@LD
	Refreshes Input Bit, 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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK00 to TK31
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, A1, A0
Clock Pulses	0.0 2s, 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	---

Area	LD operand bit
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

LD is used for the first normally open bit from the bus bar or for the first normally open bit of a logic block. If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status of the Basic Input Unit's input terminal is read and used.

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 minus 1. 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.

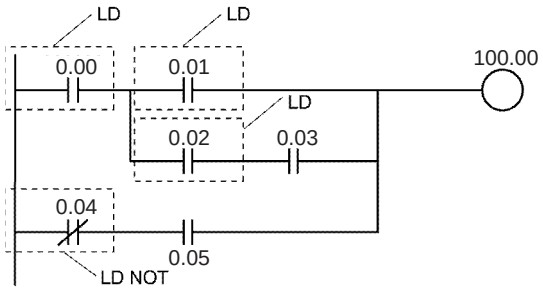
Precautions

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.

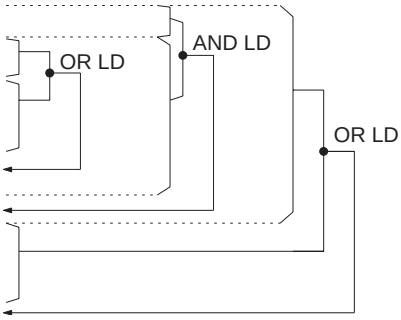
Immediate refreshing (!) can be specified for LD. An immediate refresh instruction updates the status of the input bit for CPU Unit built-in inputs just before the instruction is executed.

For LD, it is possible to combine immediate refreshing and up or down differentiation (!@ or !%). If either of these is specified, the built-in input is refreshed from the CPU Unit just before the instruction is executed and the execution condition is turned ON for one cycle only after the status goes from OFF to ON, or from ON to OFF.

Example



Instruction	Operand
LD	0.00
LD	0.01
LD	0.02
AND	0.03
OR LD	---
AND LD	---
LD NOT	0.04
AND	0.05
OR LD	---
OUT	100.00



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
Immediate Refreshing Specification		!LD NOT
Combined Variations	**Refreshes Input Bit, Restarts Logic, and Creates ON Once for Upward Differentiation**	!@LD NOT
	Refreshes Input Bit, 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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK00 to TK31
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.0 2s, 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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

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. If there is no immediate refreshing specification, the specified bit in I/O memory is read and reversed. If there is an immediate refreshing specification, the status of the Basic Input Unit's input terminal is read, reversed, and used.

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.

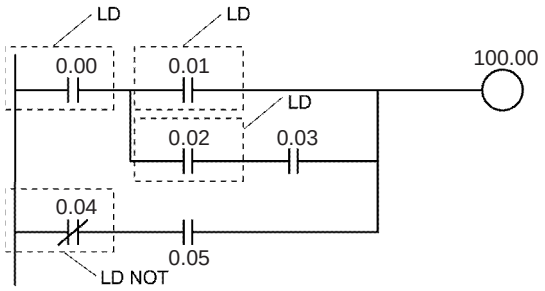
Flags

There are no flags affected by this instruction.

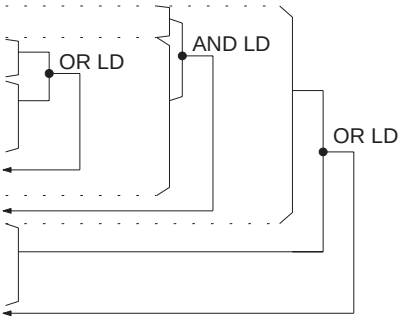
Precautions

Immediate refreshing (!) can be specified for LD NOT. An immediate refresh instruction updates the status of the input bit for a CPU Unit built-in input just before the instruction is executed.

Example



Instruction	Operand
LD	0.00
LD	0.01
LD	0.02
AND	0.03
OR LD	---
AND LD	---
LD NOT	0.04
AND	0.05
OR LD	---
OUT	100.00



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
Immediate Refreshing Specification		!AND
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation**	!@AND
	Refreshes Input Bit 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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK00 to TK31
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 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 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. If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status of the CPU Unit's input terminal is read.

Flags

There are no flags affected by this instruction.

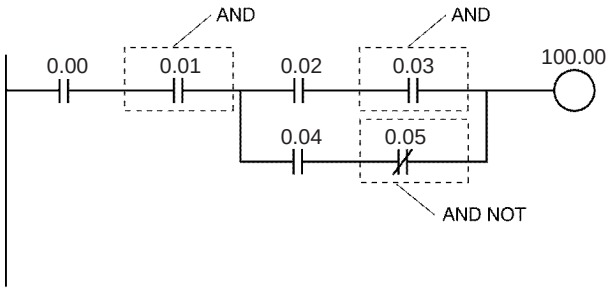
Precautions

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.

Immediate refreshing (!) can be specified for AND. An immediate refresh instruction updates the status of the input bit for CPU Unit built-in inputs just before the instruction is executed.

For AND, it is possible to combine immediate refreshing and up or down differentiation (!@ or !%). If either of these is specified, the input is refreshed from the CPU Unit just before the instruction is executed and the execution condition is turned ON for one cycle only after the status goes from OFF to ON, or from ON to OFF.

Example



Instruction	Operand
LD	0.00
AND	0.01
LD	0.02
AND	0.03
LD	0.04
AND NOT	0.05
OR LD	---
AND LD	---
OUT	100.00

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
Immediate Refreshing Specification		!AND NOT
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation**	!@AND NOT
	Refreshes Input Bit and 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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK00 to TK31
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER

Area	AND NOT bit operand
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 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 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. If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status the CPU Unit's input terminals is read.

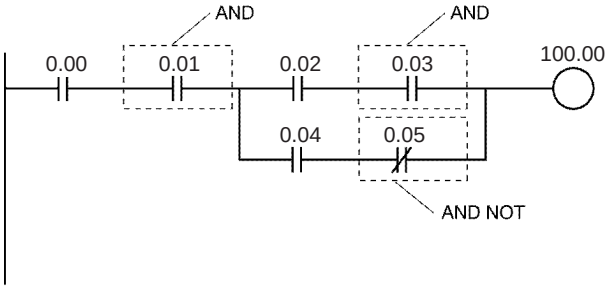
Flags

There are no flags affected by this instruction.

Precautions

Immediate refreshing (!) can be specified for AND NOT. An immediate refresh instruction updates the status of the input bit for CPU Unit built-in inputs just before the instruction is executed.

Example



Instruction	Operand
LD	0.00
AND	0.01
LD	0.02
AND	0.03
LD	0.04
AND NOT	0.05
OR LD	---
AND LD	---
OUT	100.00

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
Immediate Refreshing Specification		!OR
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation**	!@OR
	Refreshes Input Bit and 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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK00 to TK31
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 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 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). If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status of the CPU Unit's input terminal is read.

Flags

There are no flags affected by this instruction.

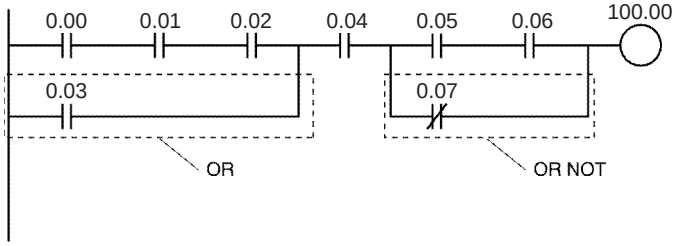
Precautions

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.

Immediate refreshing (!) can be specified for OR. An immediate refresh instruction updates the status of the input bit for a CPU Unit built-in input just before the instruction is executed.

For OR, it is possible to combine immediate refreshing and up or down differentiation (!@ or !%). If either of these is specified, the input is refreshed from the CPU Unit just before the instruction is executed and the execution condition is turned ON for one cycle only after the status of the operand bit goes from OFF to ON, or from ON to OFF.

Example



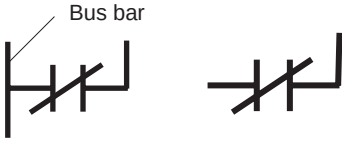
Instruction	Operand
LD	0.00
AND	0.01
AND	0.02
OR	0.03
AND	0.04
LD	0.05
AND	0.06
OR NOT	0.07
AND LD	---
OUT	100.00

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
Immediate Refreshing Specification		!OR NOT
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation**	!@OR NOT
	Refreshes Input Bit and 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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK00 to TK31
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, A1, A0
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 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 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). If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status of the CPU Unit's input terminal is read.

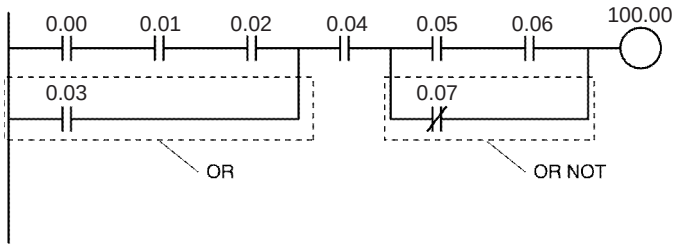
Flags

There are no flags affected by this instruction.

Precautions

Immediate refresh (!) can be specified for OR NOT. An immediate refresh instruction updates the status of the input bit from a CPU Unit built-in input just before the instruction is executed.

Example



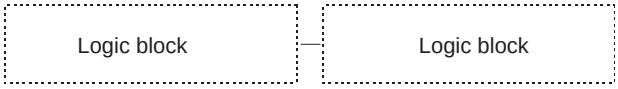
Instruction	Operand
LD	0.00
AND	0.01
AND	0.02
OR	0.03
AND	0.04
LD	0.05
AND	0.06
OR NOT	0.07
AND LD	---
OUT	100.00

3-2-7 AND LOAD: AND LD

Purpose

Takes a logical AND between logic blocks.

Ladder Symbol



Variations

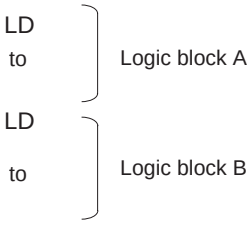
Variations	Creates ON Each Cycle AND Result is ON	AND LD
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Description

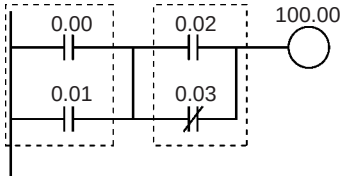
AND LD connects in series the logic block just before this instruction with another logic block.



AND LD ····· Serial connection between logic block A and logic block B.

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 0.00 or CIO 0.01 is ON) **and** either of the execution conditions in the right logic block is ON (i.e., when either CIO 0.02 is ON or CIO 0.03 is OFF).



Flags

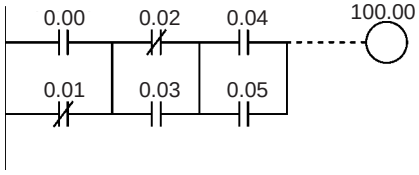
There are no flags affected by this instruction.

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	0.00
OR NOT	0.01
LD NOT	0.02
OR	0.03
AND LD	---
LD	0.04
OR	0.05
AND LD	---
. .	. .
OUT	100.00

Coding Example (2)

Instruction	Operand
LD	0.00
OR NOT	0.01
LD NOT	0.02
OR	0.03
LD	0.04
OR	0.05
. .	. .

Instruction	Operand
AND LD	---
AND LD	---
. .	. .
OUT	100.00

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.

Coding

Address	Instruction	Operand
000000	LD	0.00
000001	OR	0.01
000002	LD	0.02
000003	OR NOT	0.03
000004	AND LD	---
000005	OUT	100.00

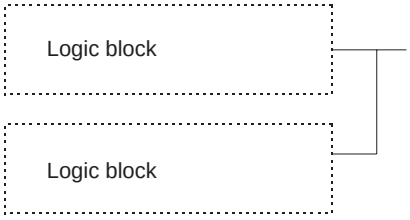
Second LD: Used for first bit of next block connected in series to previous block.

3-2-8 OR LOAD: OR LD

Purpose

Takes a logical OR between logic blocks.

Ladder Symbol



Variations

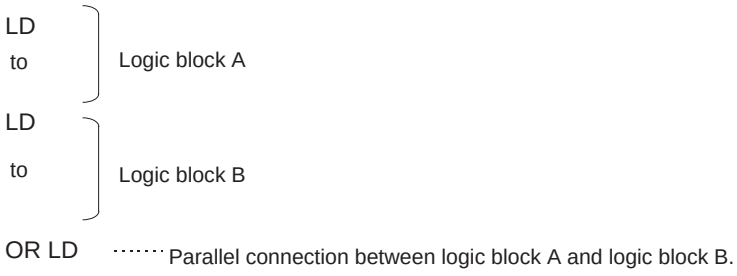
Variations	Creates ON Each Cycle AND Result is ON	OR LD
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

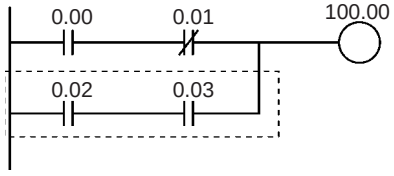
Description

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 0.00 is ON and CIO 0.01 is OFF or when CIO 0.02 and CIO 0.03 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.



Flags

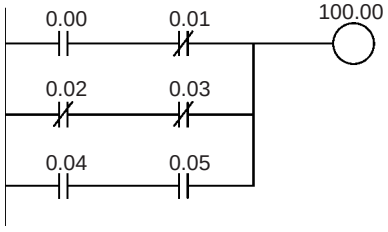
There are no flags affected by this instruction.

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	0.00
AND NOT	0.01
LD NOT	0.02
AND NOT	0.03
OR LD	---

Instruction	Operand
LD	0.04
AND	0.05
OR LD	---
. .	. .
OUT	100.00

Coding Example (2)

Instruction	Operand
LD	0.00
AND NOT	0.01
LD NOT	0.02
AND NOT	0.03
LD	0.04
AND	0.05
. .	. .
OR LD	---
OR LD	---
. .	. .
OUT	100.01

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 CX-Programmer.

Coding

Address	Instruction	Operand
000100	LD	0.00
000101	AND NOT	0.01
000102	LD	0.02
000103	AND	0.03
000104	OR LD	---
000105	OUT	100.00

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

3-2-9 Differentiated and Immediate Refreshing Instructions

The LOAD, AND, and OR instructions have differentiated and immediate refreshing variations in addition to their ordinary forms, and there are also two combinations available.

The LOAD NOT, AND NOT, OR NOT, OUT, and OUT NOT instructions have immediate refreshing variations in addition to their ordinary forms.

The I/O timing for data handled by instructions differs for ordinary and differentiated instructions, immediate refreshing instructions, and immediate refreshing 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 CPU's internal memory and CPU Unit built-in I/O, CPM1A Expansion Units, and CPM1A Expansion I/O Units.

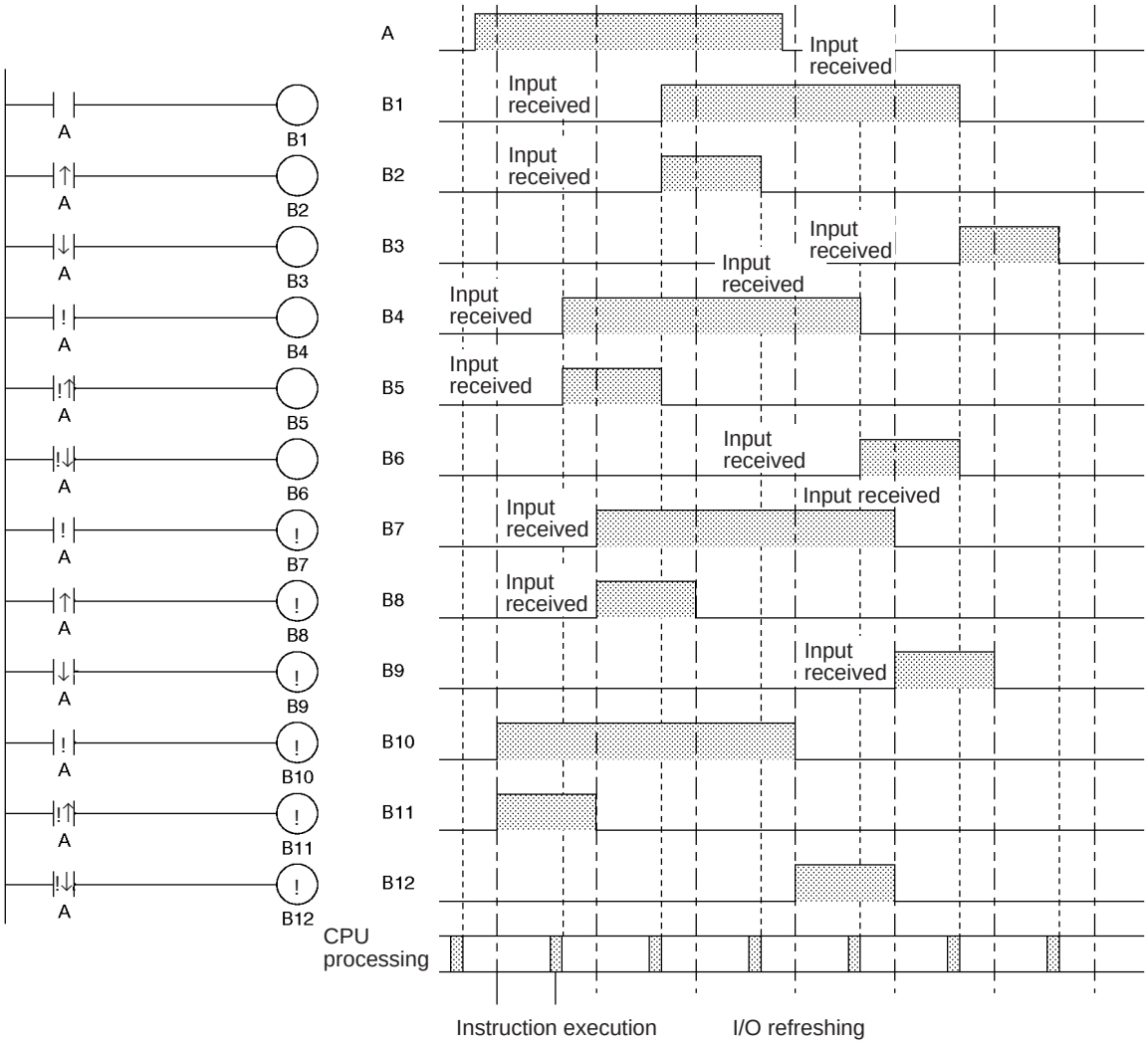
In addition to the above I/O refreshing, an immediate refresh instruction exchanges data with the I/O Unit for those words that are accessed by the instruction. An immediate refresh instruction refreshes all of the bits in the word containing the specified bit.

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 CPU 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.	
Immediate refresh	!LD, !AND, !OR, !LD NOT, !AND NOT, !OR NOT	The input data for the specified bit is taken by the CPU and the instruction is executed.	Before instruction execution
	!OUT, !OUT NOT	After the instruction is executed, the data for the specified bit is output.	After instruction execution
Differentiated up / immediate refresh	!@LD, !@AND, !@OR	The input data for the specified bit is refreshed by the CPU, and the instruction is executed once when the bit turns from OFF to ON and the ON state is held for one cycle.	Before instruction execution
Differentiated down / immediate refresh	!%LD, !%AND, !%OR	The input data for the specified bit is refreshed by the CPU, and the instruction is executed once when the bit turns from ON to OFF and the ON state is held for one cycle.	

Note Immediate refresh instructions (i.e., instructions with !) can be used only for built-in I/O on the CPU Unit. They cannot be used for I/O on CPM1A Expansion Units or CPM1A Expansion I/O Units. Use IORF(097) for I/O on CPM1A Expansion Units or CPM1A Expansion I/O Units.

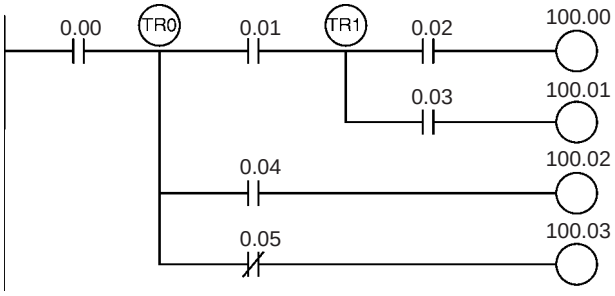
3-2-10 Operation Timing for I/O Instructions

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.

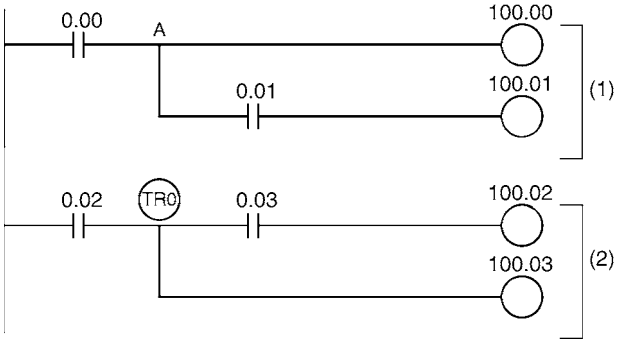


Address	Instruction	Operands
00200	LD	0.00
00201	OUT	TR0
00202	AND	0.01
00203	OUT	TR1
00204	AND	0.02
00205	OUT	100.00
00206	LD	TR1
00207	AND	0.03
00208	OUT	100.01
00209	LD	TR0
00210	AND	0.04
00211	OUT	100.02
00212	LD	TR0
00213	AND NOT	0.05
00214	OUT	100.03

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.



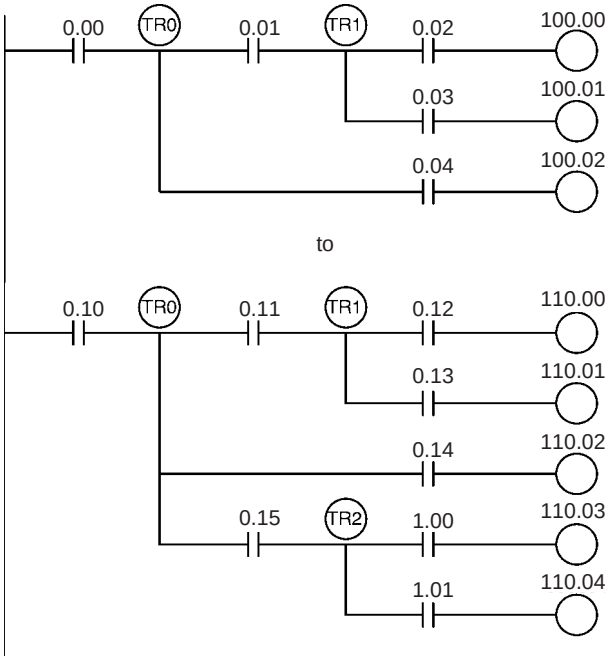
In instruction block (1), the ON/OFF status at point A is the same as for output CIO 100.00, so AND 0.01 and OUT 100.01 can be coded without requiring a TR bit. In instruction block (2), the status of the branching point and that of output CIO 100.02 are not necessarily the same, so a TR bit must be 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.

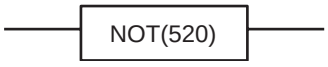


3-2-12 NOT: NOT(520)

Purpose

Reverses the execution condition.

Ladder Symbol



Variations

Variations	Reverses the Execution Condition Each Cycle	NOT(520)
Immediate Refreshing Specification		Not supported

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

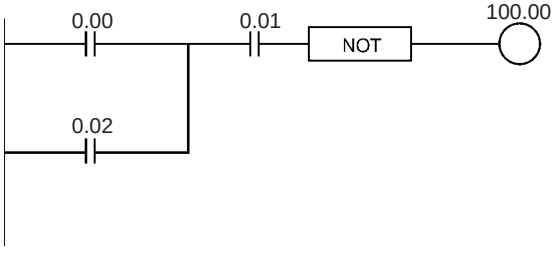
There are no flags affected by NOT(520)

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.

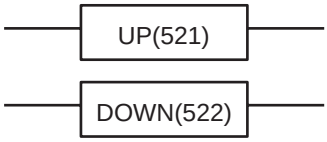
Input bit status			Output bit status
CIO 0.00	**CIO 0.01**	**CIO 0.02**	**CIO 0.03**
1	1	1	0
1	1	0	0
1	0	1	1
0	1	1	0
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	1

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)
Immediate Refreshing Specification		Not supported

Variations	**Creates ON Once for Downward Differentiation**	UP(522)
Immediate Refreshing Specification		Not supported

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).

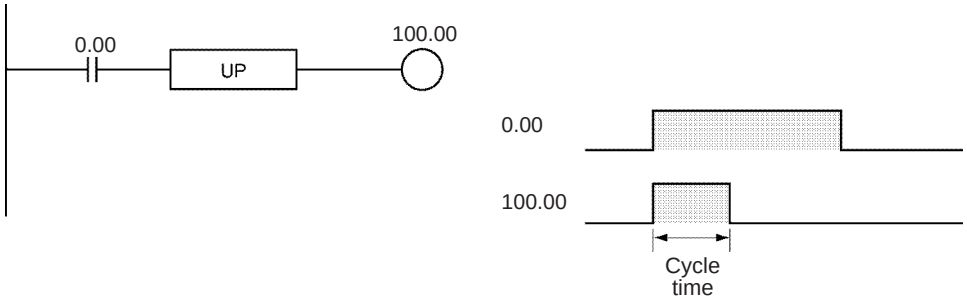
Precautions

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).

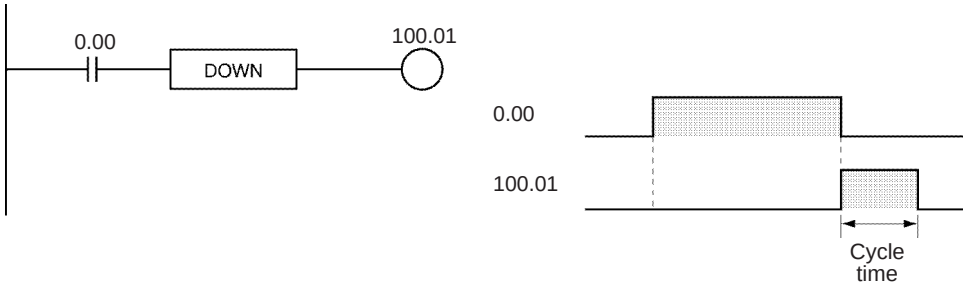
The operation of UP(521) and DOWN(522) depends on the execution condition for the instruction as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped program section, or a subroutine. Refer to *3-4-4 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 3-4-6 JUMP and JUMP END: JMP(004) and JME(005)*, and *3-19 Interrupt Control Instructions* for details.

Examples

When CIO 0.00 goes from OFF to ON in the following example, CIO 100.00 is turned ON for just one cycle.



When CIO 0.00 goes from ON to OFF in the following example, CIO 100.01 is turned ON for just one cycle.



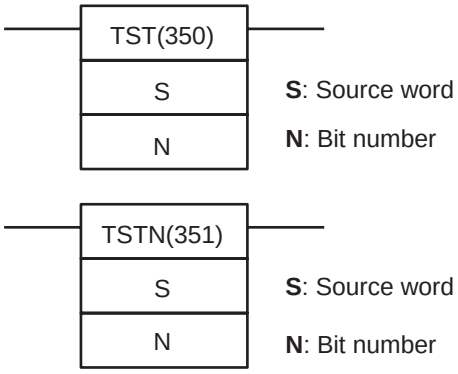
3-2-14 BIT TEST: TST(350) and TSTN(351)

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)
Immediate Refreshing Specification		Not supported

Variations	Executed Each Cycle	TSTN(351)
Immediate Refreshing Specification		Not supported

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 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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	OFF or unchanged
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

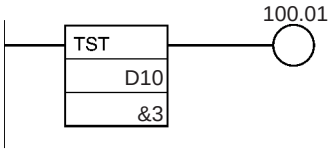
Precautions

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).

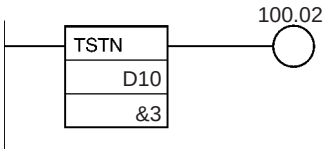
Examples

LD TST(350) and LD TSTN(351)

In the following example, CIO 100.01 is turned ON when bit 3 of D10 is ON.

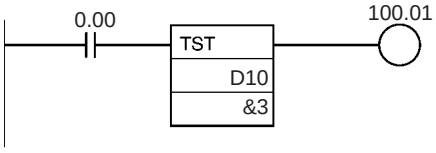


In the following example, CIO 100.02 is turned ON when bit 3 of D10 is OFF.

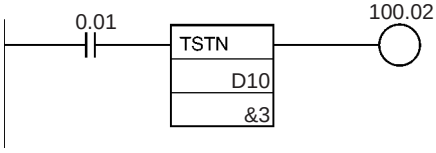


AND TST(350) and AND TSTN(351)

In the following example, CIO 100.01 is turned ON when CIO 0.00 and bit 3 of D10 are both ON.

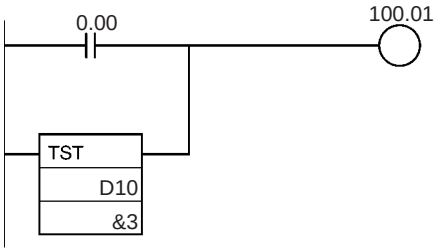


In the following example, CIO 100.02 is turned ON when CIO 0.01 is ON and bit 3 of D10 is OFF.

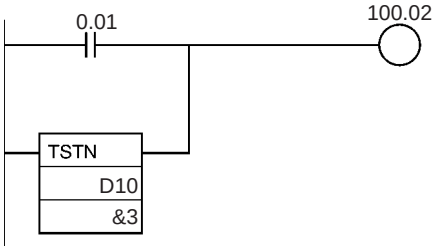


OR TST(350) and OR TSTN(351)

In the following example, CIO 100.01 is turned ON when CIO 0.00 or bit 3 of D10 is ON.



In the following example, CIO 100.02 is turned ON when CIO 0.01 is ON or bit 3 of D10 is OFF.



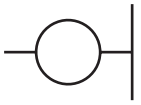
3-3 Sequence Output Instructions

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.
Immediate Refreshing Specification		!OUT

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 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
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 Registers	---

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 ,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

If there is no immediate refreshing specification, the status of the execution condition (power flow) is written to the specified bit in I/O memory. If there is an immediate refreshing specification, the status of the execution condition (power flow) is also written to the CPU Unit's output terminal in addition to the output bit in I/O memory.

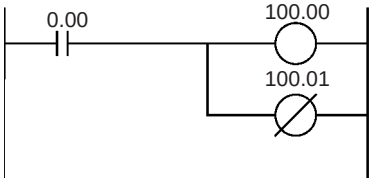
Flags

There are no flags affected by this instruction.

Precautions

Immediate refreshing (!) can be specified for OUT and OUT NOT. An immediate refresh instruction updates the status of the output terminal on the CPU Unit just after the instruction is executed at the same time as it writes the status of the execution condition (power flow) to the specified output bit in I/O memory.

Example



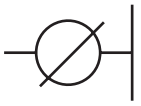
Instruction	Operand
LD	0.00
OUT	100.00
OUT NOT	100.01

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.
Immediate Refreshing Specification		!OUT NOT

Applicable Program Areas

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

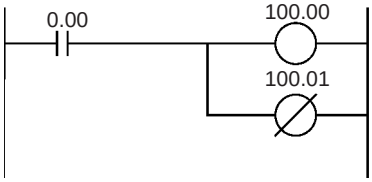
Operand Specifications

Area	OUT bit operand
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
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 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 ,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description If there is no immediate refreshing specification, the status of the execution condition (power flow) is reversed and written to a specified bit in I/O memory. If there is an immediate refreshing specification, the status of the execution condition (power flow) is reversed and also written to the CPU Unit's output terminal in addition to the output bit in I/O memory.

Flags There are no flags affected by this instruction.

Example

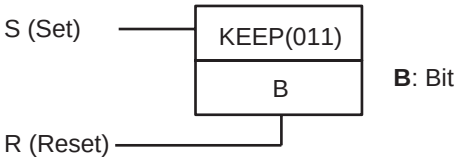


Instruction	Operand
LD	0.00
OUT	0.01
OUT NOT	0.02

3-3-3 KEEP: KEEP(011)

Purpose Operates as a latching relay.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	KEEP(011)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		!KEEP(011)

Applicable Program Areas

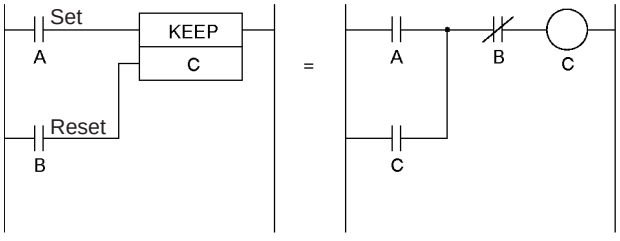
Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

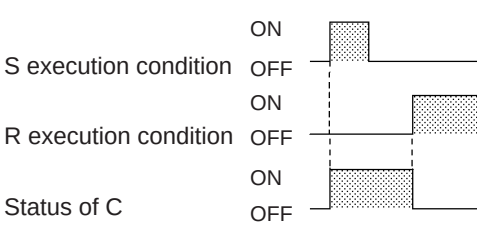
Operand Specifications

Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15

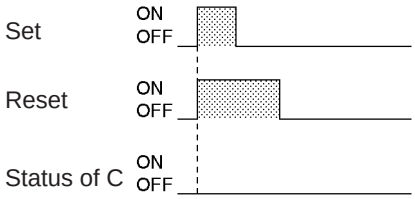
Description

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.

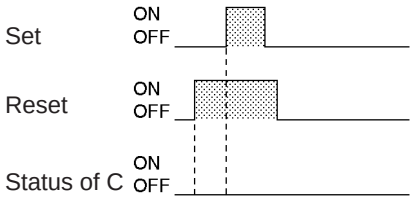




If S and R are ON simultaneously, the reset input takes precedence.

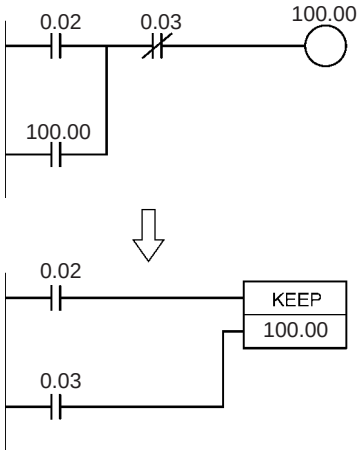


The set input (S) cannot be received while R is ON.

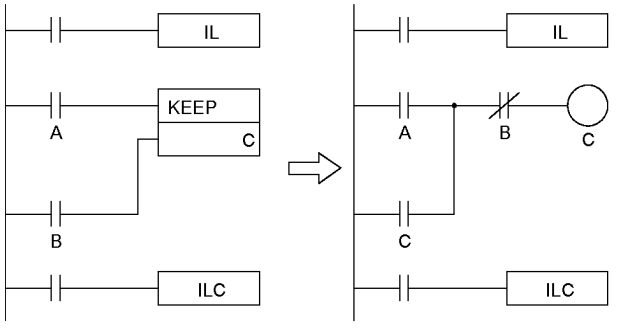


KEEP(011) has an immediate refreshing variation (!KEEP(011)). When an external output bit has been specified for B in a !KEEP(011) instruction, any changes to B will be refreshed when !KEEP(011) is executed and reflected immediately in the output bit for the CPU Unit built-in output.

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



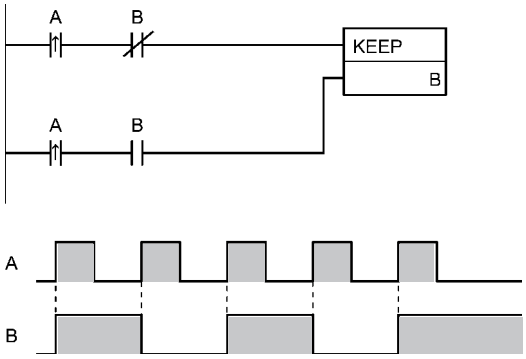
Self-maintaining bits programmed with KEEP(011) will maintain status even in an interlock program section, unlike the self-maintaining bit programmed without KEEP(011).



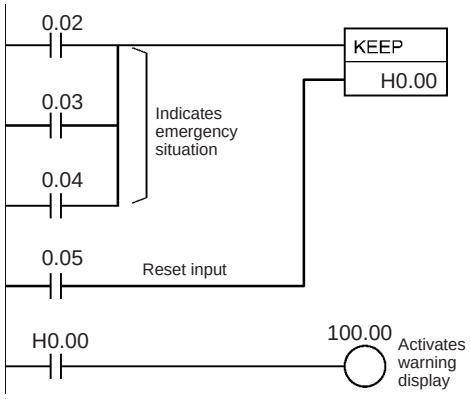
Output bit C will maintain its previous status in an interlock.

Output bit C will be turned OFF in an interlock.

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



If a holding bit is used for B, the bit status will be retained even during a power interruption. KEEP(011) can thus be used to program bits that will maintain status after restarting the PLC following a power interruption. An example of this that can be used to produce a warning display following a system shut-down for an emergency situation is shown below.



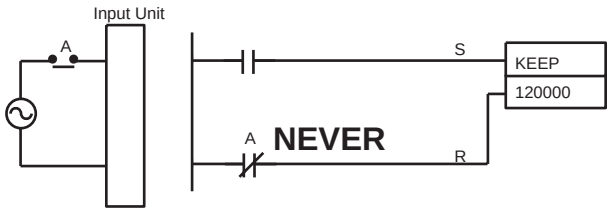
The status of I/O Area bits can be retained in the event of a power interruption by turning ON the IOM Hold Bit and setting IOM Hold Bit Hold in the PLC Setup. In this case, I/O Area bits used in KEEP(011) will maintain status after restarting the PLC following a power interruption, just like holding bits. Be sure to restart the PLC after changing the PLC Setup; otherwise the new settings will not be used.

Flags

No flags are affected by KEEP(011).

Precautions

Never use an input bit in a normally closed condition on the reset (R) for KEEP(011) when the input device uses an AC power supply. The delay in shutting down the PLC's DC power supply (relative to the AC power supply to the input device) can cause the operand bit of KEEP(011) to be reset. This situation is shown below.

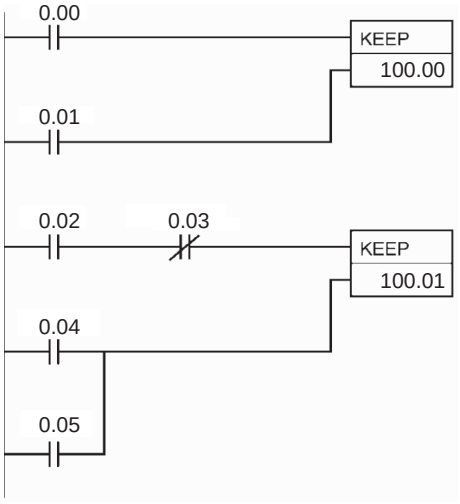


The operands for KEEP(011) are input in a different order in ladder diagrams and mnemonic code.
Ladder diagram order: Set input → KEEP(011) → Reset input
Mnemonic code order: Set input → Reset input → KEEP(011)

Example

When CIO 0.00 goes ON in the following example, CIO 100.00 is turned ON. CIO 100.00 remains ON until CIO 0.01 goes ON.

When CIO 0.02 goes ON and CIO 0.03 goes OFF in the following example, CIO 100.01 is turned ON. CIO 100.01 remains ON until CIO 0.04 or CIO 0.05 goes ON.



Coding

Address	Instruction	Operand
000100	LD	0.00
000101	LD	0.01
000102	KEEP (011)	100.00
000103	LD	0.02
000104	AND NOT	0.03
000105	LD	0.04
000106	OR	0.05
000107	KEEP (011)	100.01

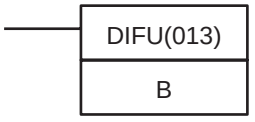
Note KEEP(011) is input in different orders on in ladder and mnemonic form. In ladder form, input the set input, KEEP(011), and then the reset input. In mnemonic form, input the set input, the reset input, and then KEEP(011).

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

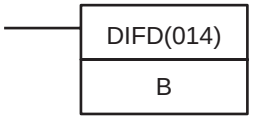
Purpose

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
Immediate Refreshing Specification		!DIFU(013)

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

Applicable Program Areas

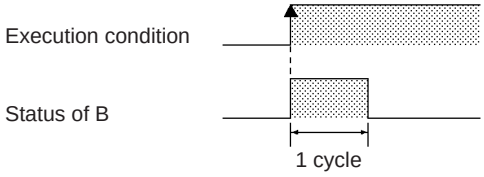
Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operand Specifications

Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to ,15–(– –) IR

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.

DIFU(013) and DIFD(014) have immediate refreshing variations (!DIFU(013) and !DIFD(014)). When an external output bit has been specified for B in one of these instructions, any changes to B will be refreshed when the instruction is executed and reflected immediately in the output bit for the CPU Unit built-in output.

UP(521) and DOWN(522) can be used to execute an instruction for just one cycle when the execution condition goes from OFF → ON or ON → OFF. Refer to *3-2-13 CONDITION ON/OFF: UP(521) and DOWN(522)* for details.

Flags

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

Precautions

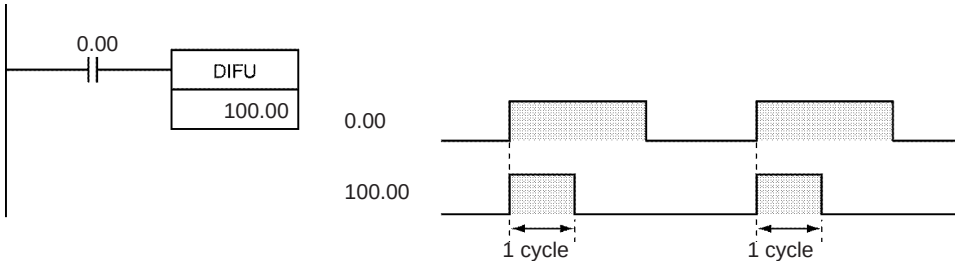
The operation of DIFU(013) or DIFD(014) depends on the execution condition for the instruction itself as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped program section, or a subroutine. Refer to *3-4-4 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 3-4-6 JUMP and JUMP END: JMP(004) and JME(005)*, and *3-19 Interrupt Control Instructions* for details.

If DIFU(013) is used in a FOR-NEXT loop and the loop repeats in a cycle, the controlled bit will be always ON or always OFF within that loop.

Examples

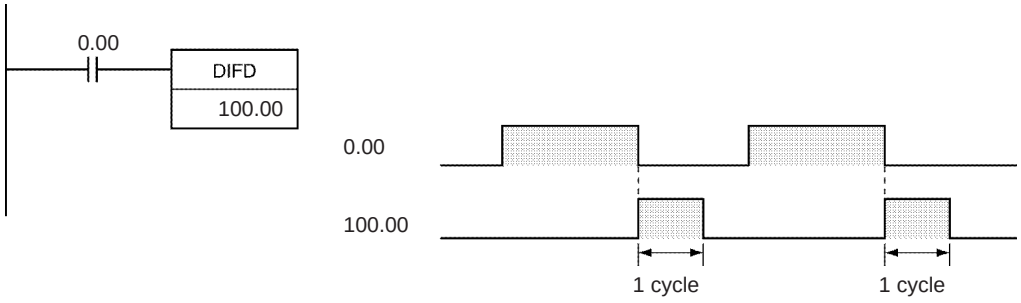
Operation of DIFU(013)

When CIO 0.00 goes from OFF to ON in the following example, CIO 100.00 is turned ON for one cycle.



Operation of DIFD(014)

When CIO 0.00 goes from ON to OFF in the following example, CIO 100.00 is turned ON for one cycle.

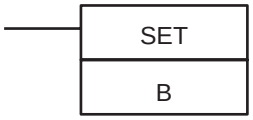


3-3-5 SET and RESET: SET and RSET

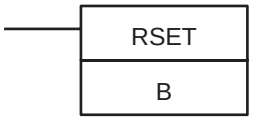
Purpose

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
Immediate Refreshing Specification		!SET
Combined variations	**Executed Once and Bit Refreshed Immediately for Upward Differentiation**	!@SET
	Executed Once and Bit Refreshed Immediately for Downward Differentiation	!%SET

Variations	**Executed Each Cycle for ON Condition**	RSET
	Executed Once for Upward Differentiation	@RSET
	Executed Once for Downward Differentiation	%RSET
Immediate Refreshing Specification		!RSET
Combined Variations	**Immediate Refreshing Once for Upward Differentiation**	!@RSET
	Immediate Refreshing Once for Downward Differentiation	!%RSET

Applicable Program Areas

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

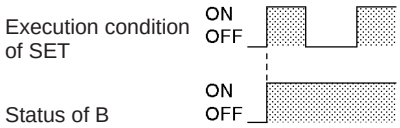
Operand Specifications

Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
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 Registers	---

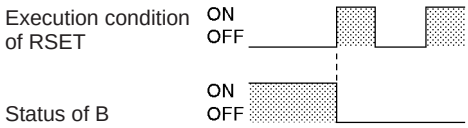
Area	B
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to ,–(– –) IR15

Description

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.



SET and RSET have immediate refreshing variations (!SET and !RSET). When an external output bit has been specified for B in one of these instructions, any changes to B will be refreshed when the instruction is executed and reflected immediately in the output bit for the CPU Unit built-in output.

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

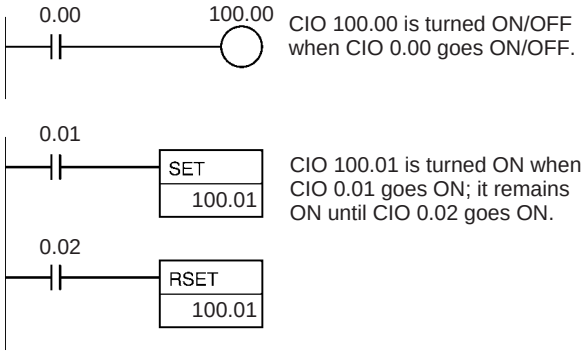
SET and RSET cannot be used to set and reset timers and counters.

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 program section is interlocked or jumped.

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.

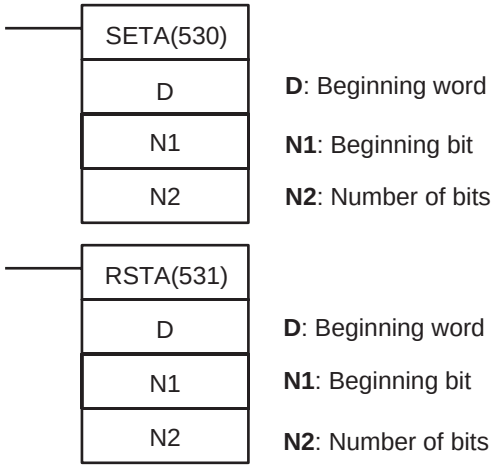


3-3-6 MULTIPLE BIT SET/RESET: SETA(530)/RSTA(531)

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	Executed Each Cycle for ON Condition	SETA(530)
	Executed Once for Upward Differentiation	@SETA(530)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Variations	Executed Each Cycle for ON Condition	RSTA(531)
	Executed Once for Upward Differentiation	@RSTA(531)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		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.

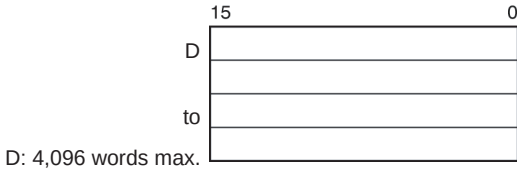
N1: Beginning Bit

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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767 (n = 0 to C)		
Indirect DM addresses in BCD	*D0 to *D32767 (n = 0 to C)		
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

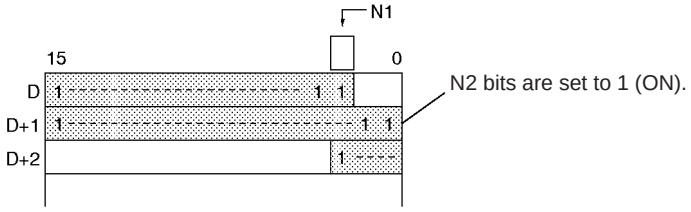
Description

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).

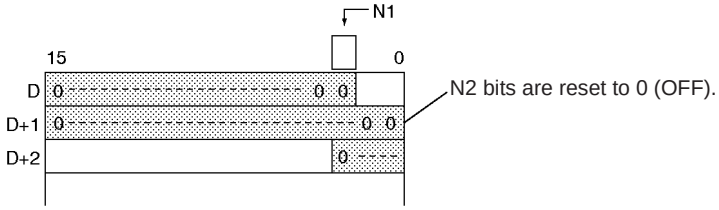


SETA(530) can be used to turn ON bits in data areas that are normally accessed by words only, such as the DM area.

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.

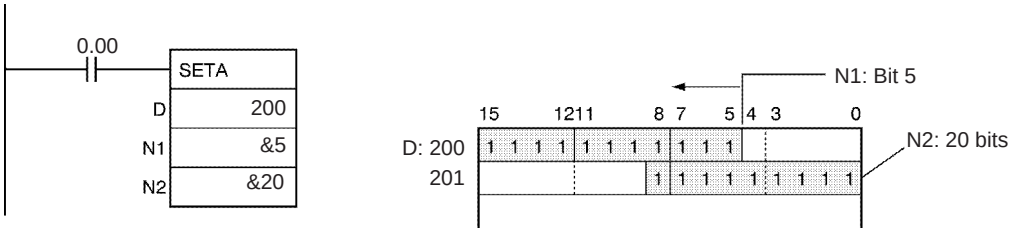
Flags

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

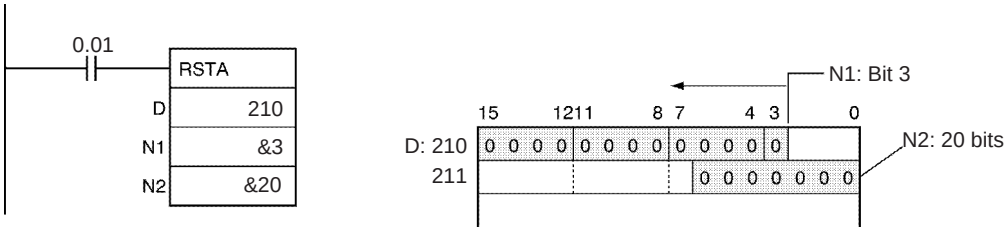
SETA(530) Example

When CIO 0.00 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 5 of CIO 200 are turned ON.



RSTA(531) Example

When CIO 0.01 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 3 of CIO 210 are turned OFF.

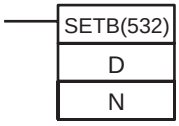


3-3-7 SINGLE BIT SET/RESET: SETB(532)/RSTB(533)

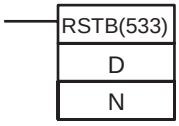
Purpose

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
Immediate Refreshing Specification		!SETB(532)
Combined Variations	**Executed Once and Bit Refreshed Immediately for Upward Differentiation**	!@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
Immediate Refreshing Specification		!RSTB(533)
Combined Variations	**Executed Once and Bit Refreshed Immediately for Upward Differentiation**	!@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

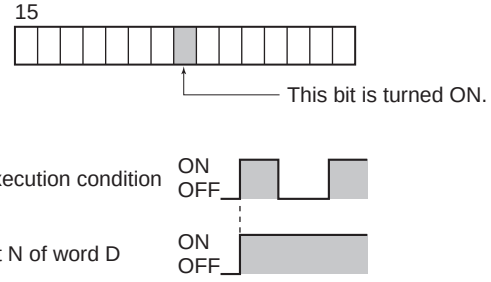
Area	D	N
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	A0 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

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

Operation of SETB(532)

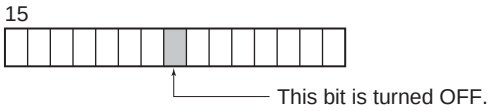
SETB(532) turns ON bit N of word D when the execution condition is ON. The status of the bit is not affected when the execution condition is OFF. Unlike SET, SETB(532) can turn ON a bit in the DM area.

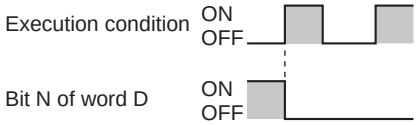


Bits turned ON by SETB(532) can be turned OFF by any other instruction, not just RSTB(533).

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 for the CPU Unit built-in output.

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

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

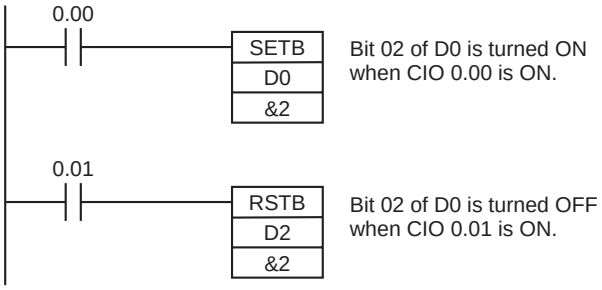
1. The instructions operate in the same way when the specified bit is in the CIO, W, H, or A Area.
2. The SETB(532) and RSTB(533) instructions can control bits in the DM Area, unlike SET and RSET.

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.

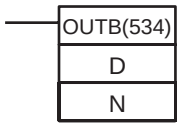


3-3-8 SINGLE BIT OUTPUT: OUTB(534)

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
Immediate Refreshing Specification		!OUTB(534)

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).

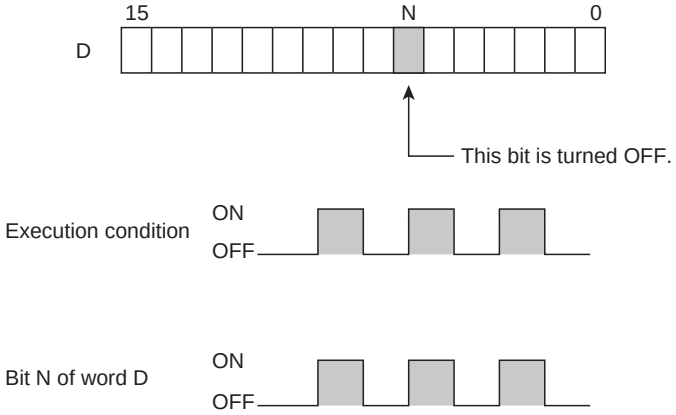
Operand Specifications

Area	D	N
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	A0 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	#0000 to #000F (binary) or &0 to &15

Area	D	N
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

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 CPU Unit's output terminal as well as the output bit in I/O memory.

Flags

There are no flags affected by this instruction.

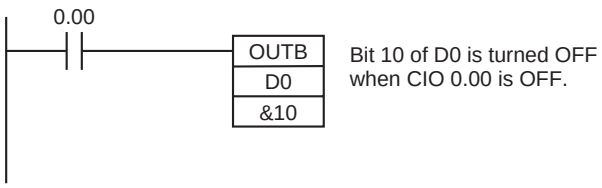
Precautions

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 CPU Unit built-in output, 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

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)
Immediate Refreshing Specification		Not supported

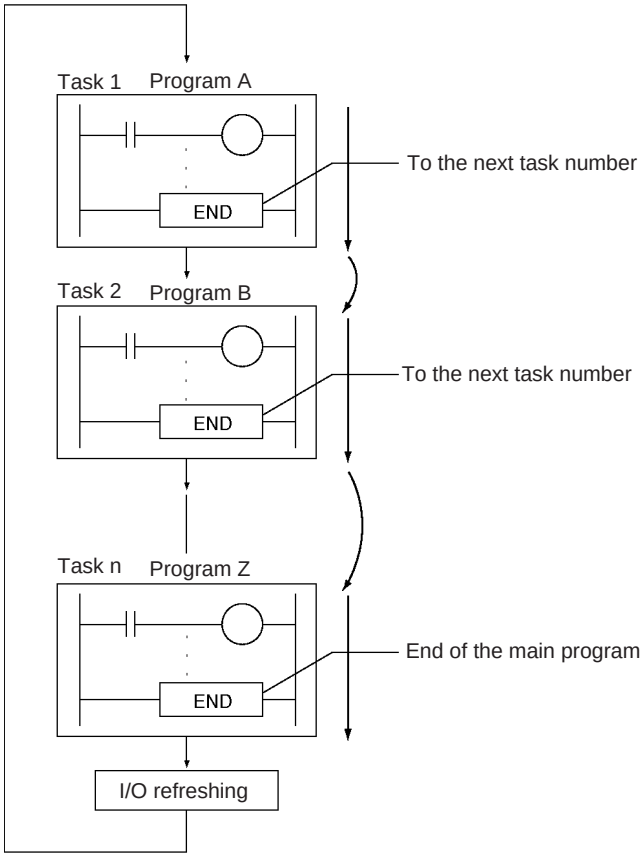
Applicable Program Areas

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)
Immediate Refreshing Specification		Not supported

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 Overview of Interlock Instructions

Interlock Instructions

The following instruction combinations can be used to interlock outputs in a program section.

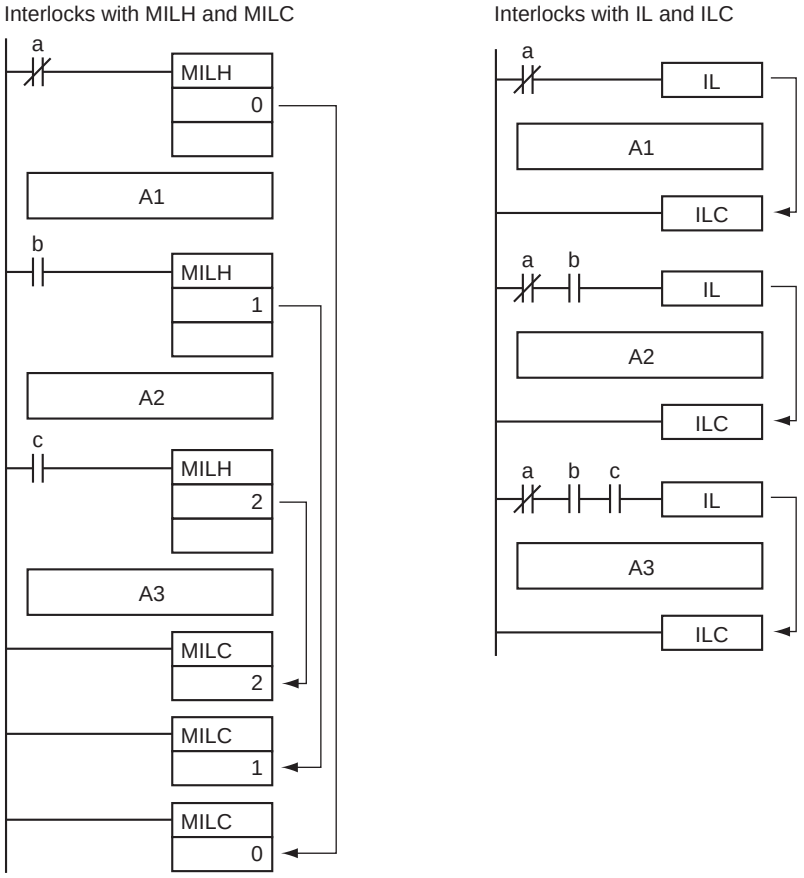
- INTERLOCK and INTERLOCK CLEAR (IL(002) and IL(003))
- MULTI-INTERLOCK DIFFERENTIATION HOLD and MULTI-INTERLOCK CLEAR (MILH(517) and MILC(519))*

  Note MILH(517) holds the status of the Differentiation Flag, so differentiated instructions that were interlocked are executed after the interlock is cleared.
- MULTI-INTERLOCK DIFFERENTIATION RELEASE and MULTI-INTERLOCK CLEAR (MILR(518) and MILC(519))*

  Note MILR(518) does not hold the status of the Differentiation Flag, so differentiated instructions that were interlocked are not executed after the interlock is cleared.

Differences between Interlocks and Multiple Interlocks

Regular interlocks (IL(002) and IL(003)) cannot be nested, but multiple interlocks (MILH(517), MILR(518), and MILC(519)) can be nested. Ladder programming can be simplified by nesting multiple interlocks, as shown in the following diagram.



Differences between MILH(517) and MILR(518)

Differentiated instructions (DIFU, DIFD, or instructions with a @ or % prefix) operate differently in interlocks created with MILH(517) and MILR(518).

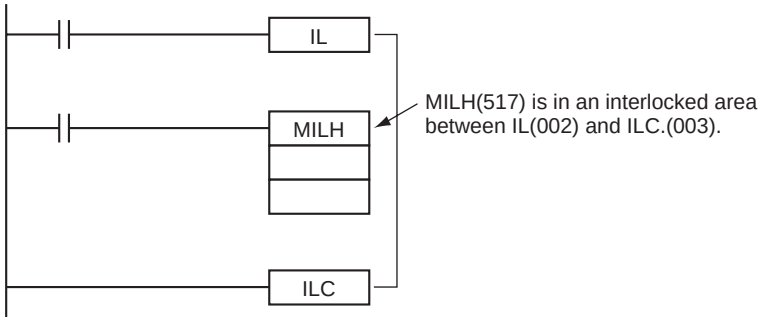
The operation of differentiated instructions in an interlock created with MILH(517) is identical to the operation in an interlock created with IL(002).

For details, refer to *3-4-5 MULTI-INTERLOCK DIFFERENTIATION HOLD, MULTI-INTERLOCK DIFFERENTIATION RELEASE, and MULTI-INTERLOCK CLEAR: MILH(517), MILR(518), and MILC(519)*.

Precautions

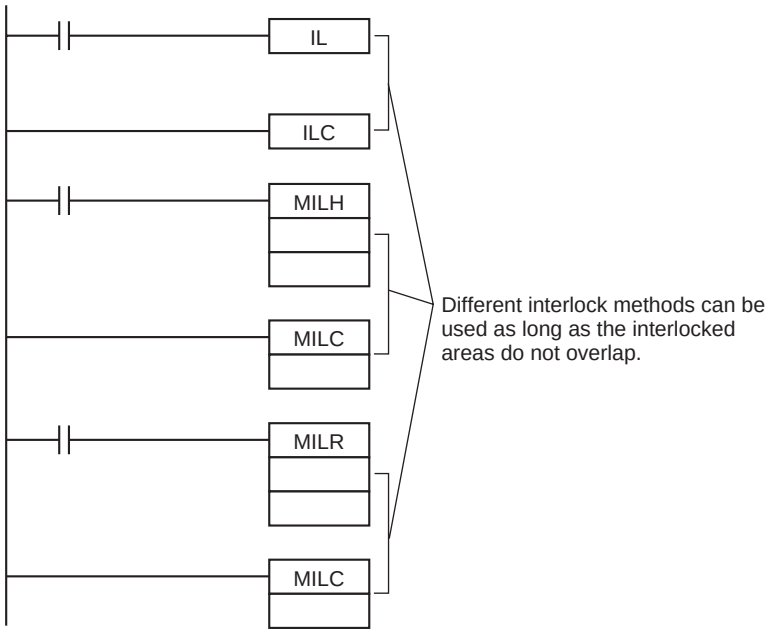
Do not combine interlocks created with different interlock instructions (IL-ILC, MILH-MILC, and MILR-MILC). The interlocks may not operate properly if different interlock methods are used together. For details on combining instructions, refer to *3-4-5 MULTI-INTERLOCK DIFFERENTIATION HOLD, MULTI-INTERLOCK DIFFERENTIATION RELEASE, and MULTI-INTERLOCK CLEAR: MILH(517), MILR(518), and MILC(519)*.

For example, an MILH(517) instruction cannot be inserted between IL(002) and IL(003).



Note The different interlocks (IL-ILC, MILH-MILC, and MILR-MILC) can be used together as long as the interlocked program sections do not overlap.

For example, all three interlock methods can be used without overlapping, as shown in the following diagram.



Differences between Interlocks and Jumps

The following table shows the differences between interlocks (created with IL(002)/ILC(003), MILH(517)/MILC(519), or MILR(518)/MILC(519)) and jumps created with JMP(004)/JME(005).

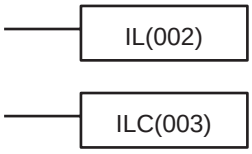
Item	Treatment in IL(002)/ILC(003), MILH(517)/ MILC(519), or MILR(518)/MILC(519))	Treatment in JMP(004)/JME(005)
Instruction execution	Instructions other than OUT, OUT NOT, OUTB(534), and timer instructions are not executed.	No instructions are executed.
Output status in instructions	Except for outputs in OUT, OUT NOT, OUTB(534), and timer instructions, all outputs retain their previous status.	All outputs retain their previous status.
Bits in OUT, OUT NOT, OUTB(534)	OFF	All outputs retain their previous status.
Status of timer instructions (except (TTIM(087), TTIMX(555), MTIM(543), and MTIMX(554))	Reset	Operating timers (TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552) only) continue timing because the PVs are updated even when the timer instruction is not being executed.

3-4-4 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)
Immediate Refreshing Specification		Not supported

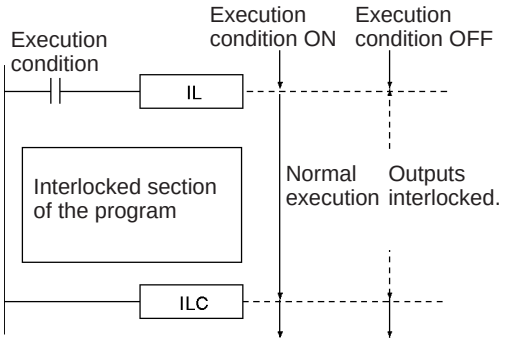
Variations	Executed Each Cycle for ON Condition	ILC(003)
Immediate Refreshing Specification		Not supported

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.



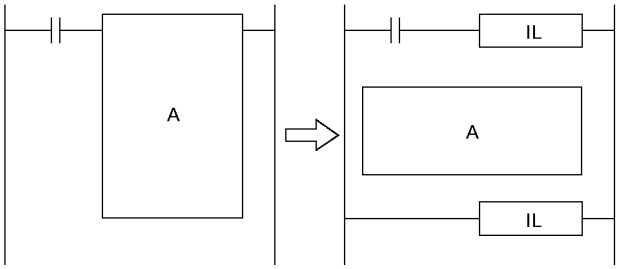
The following table shows the treatment of various outputs in an interlocked section between IL(002) and ILC(003).

Instruction		Treatment
Bits specified in OUT, OUT NOT, or OUTB(534)		OFF
TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TIML(542), and TIMXL(553)	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 TTIM(087), TTIMX(555), MTIM(543), MTIMX(554), SET, RSET, CNT, CNTX(546), CNTR(012), CNTRX(548), SFT, and KEEP(011) retain their previous status.

If there are bits which you want to remain 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, OUTB(534), and timer instructions are not executed.	No instructions are executed.
Output status in instructions	Except for outputs in OUT, OUT NOT, OUTB(534), and timer instructions, all outputs retain their previous status.	All outputs retain their previous status.
Bits in OUT, OUT NOT, OUTB(534)	OFF	All outputs retain their previous status.
Status of timer instructions (except (TTIM(087), TTIMX(555), MTIM(543), and MTIMX(554))	Reset	Operating timers (TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552) only) continue timing because the PVs are updated even when the timer instruction is not being executed.

Flags

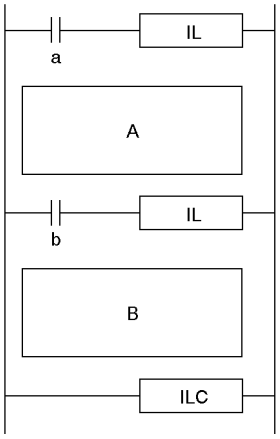
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

Precautions

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

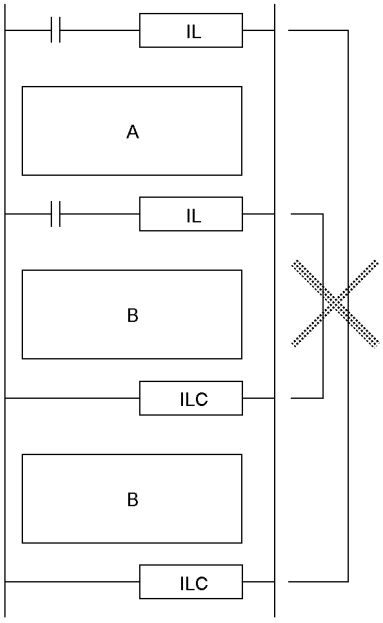
The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between IL(002) and ILC(003). Changes in the execution condition for DIFU(013), DIFD(014), or a differentiated instruction are not recorded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

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

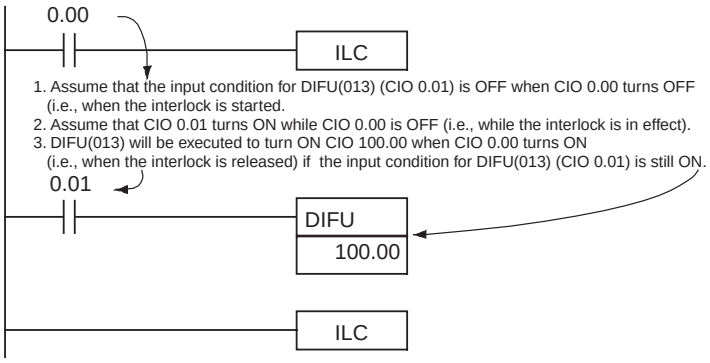
IL(002) and ILC(003) cannot be nested, as in the following diagram. (Use MILH(517)/MILR(518) and MILC(519) when it is necessary to nest interlocks.)



Differential Instruction in Interlocks

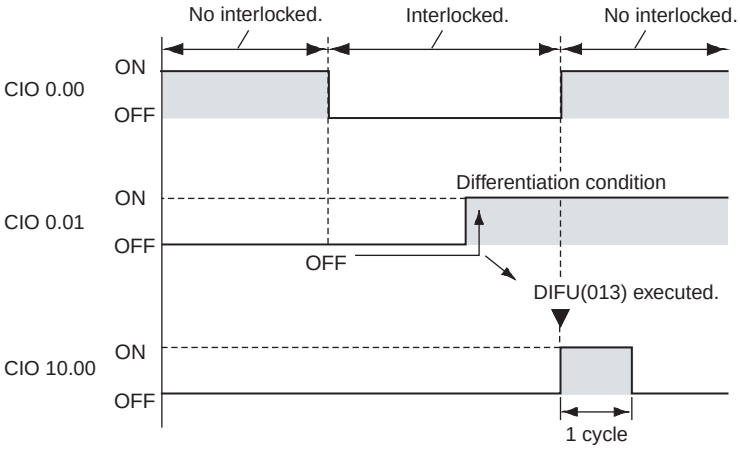
Differentiated instructions (DIFU(013), DIFD(014), or instructions with a @ or % prefix) written between IL(002) and ILC(003) are executed according to changes in memory status between when the interlock is started and when it is released. If a differentiated condition is met, it will be effected when the interlock is released.

For example, if the input condition for DIFU(013) is OFF when an interlock is started and ON when the interlock is released, the operand bit of DIFU(013) will be turned ON when the interlock is released.



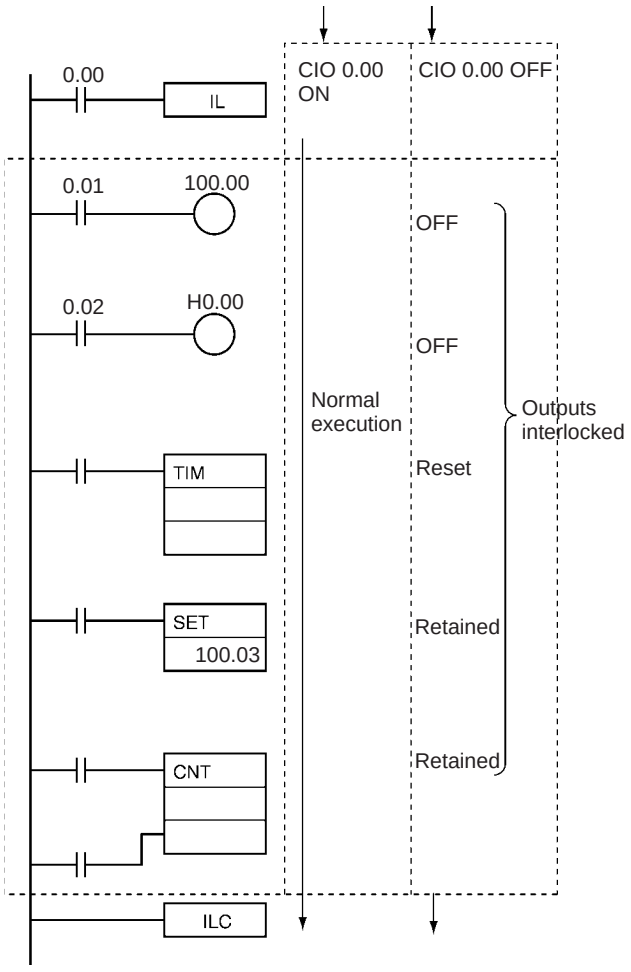
IL(002) affects differentiated operation in the same way as MILH(517).

Timing Chart



Examples

When CIO 0.00 is OFF in the following example, all outputs between IL(002) and ILC(003) are interlocked. When CIO 0.00 is ON in the following example, the instructions between IL(002) and ILC(003) are executed normally.



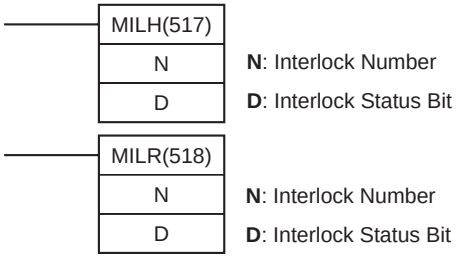
3-4-5 MULTI-INTERLOCK DIFFERENTIATION HOLD, MULTI-INTERLOCK DIFFERENTIATION RELEASE, and MULTI-INTERLOCK CLEAR: MILH(517), MILR(518), and MILC(519)

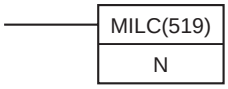
Purpose

Interlocks all outputs between MILH(517) (or MILR(518)) and MILC(519) when the execution condition for MILH(517) (or MILR(518)) is OFF. MILH(517) (or MILR(518)) and MILC(519) are normally used in pairs.

Unlike the IL(002)/ILC(003) interlocks, the MILH(517)/MILC(519) and MILR(518)/MILC(519) interlocks can be nested. The operation of differentiated instructions is different for interlocks created with MILH(517) and MILR(518).

Ladder Symbols





N: Interlock Number

Operands

N: Interlock Number

The interlock number must be between 0 and 15. Match the interlock number of the MILH(517) (or MILR(518)) instruction with the same number in the corresponding MILC(519) instruction.

The interlock numbers can be used in any order.

D: Interlock Status Bit

- ON when the program section is not interlocked.
- OFF when the program section is interlocked.

When the interlock is engaged, the Interlock Status Bit can be force-set to release the interlock. Conversely, when the interlock is not engaged, the Interlock Status Bit can be force-reset to engage the interlock.

Operand Specifications

Area	N	D
CIO Area	---	CIO 0.00 to CIO 6143.15
Work Area	---	W0.00 to W511.15
Holding Bit Area	---	H0.00 to H511.15
Auxiliary Bit Area	---	A0.00 to A959.15
Timer Area	---	---
Counter Area	---	---
DM Area	---	---
Indirect DM addresses in binary	---	---
Indirect DM addresses in BCD	---	---
Constants	0 to 15	---
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 IR15

Variations

Variations	Interlocks when OFF/Does Not interlock when ON	MILH(517) and MILR(518)
Immediate Refreshing Specification		Not supported

Variations	Executed Each Cycle for ON Condition	MILC(519)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

The following table shows the applicable program areas for MILH(517), MILR(518), and MILC(519).

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

Description

When the execution condition for MILH(517) (or MILR(518)) with interlock number N is OFF, the outputs for all instructions between that MILH(517)/MILR(518) instruction and the next MILC(519) with interlock number N are interlocked.

When the execution condition for MILH(517) (or MILR(518)) with interlock number N is ON, the instructions between that MILH(517)/MILR(518) instruction and the next MILC(519) with interlock number N are executed normally.

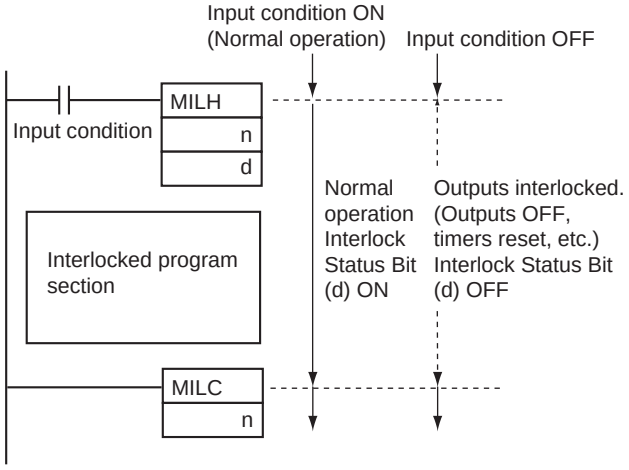
Interlock Status

The following table shows the treatment of various outputs in an interlocked section between MILH(517)/MILR(518) instruction and the next MILC(519).

Instruction		Treatment
Bits specified in OUT, OUT NOT, or OUTB(534)		OFF
TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TIML(542), and TIMXL(553)	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 TTIM(087), TTIMX(555), MTIM(543), MTIMX(554), SET, RSET, CNT, CNTX(546), CNTR(012), CNTRX(548), SFT, and KEEP(011) retain their previous status.

The MILH(517)/MILR(518) instruction turns OFF the Interlock Status Bit (operand D) when the interlock is in engaged and turns ON the bit when the interlock is not engaged. Consequently, the Interlock Status Bit can be monitored to check whether or not the interlock for a given interlock number is engaged.



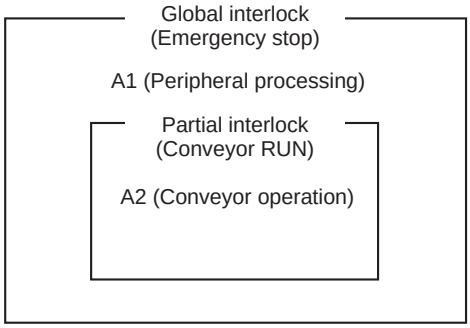
Nesting

Interlocks are nested when an interlocked program section (MILH(517)/MILR(518) and MILC(519) combination) is placed within another interlocked program section (MILH(517)/MILR(518) and MILC(519) combination). Interlocks can be nested up to 16 levels.

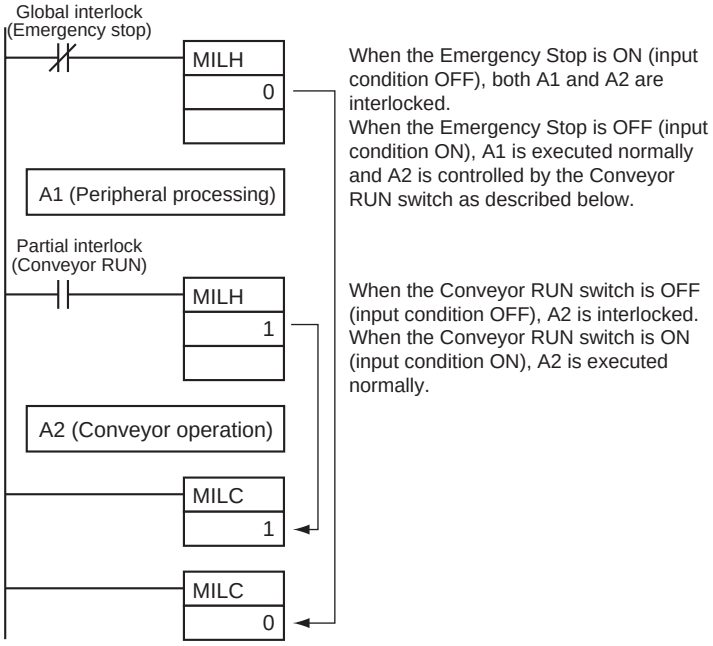
Nesting can be used for the following kinds of applications.

• Example 1

Interlocking the entire program with one condition and interlocking a part of the program with another condition (1 nesting level)

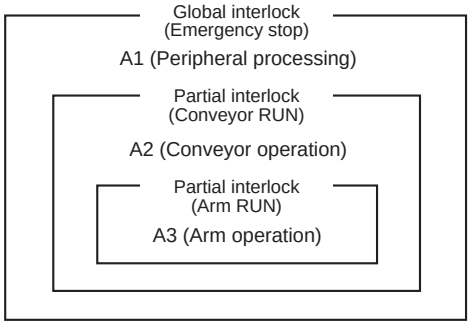


- A1 and A2 are interlocked when the Emergency Stop Button is ON.
- A2 is interlocked when Conveyor RUN is OFF.



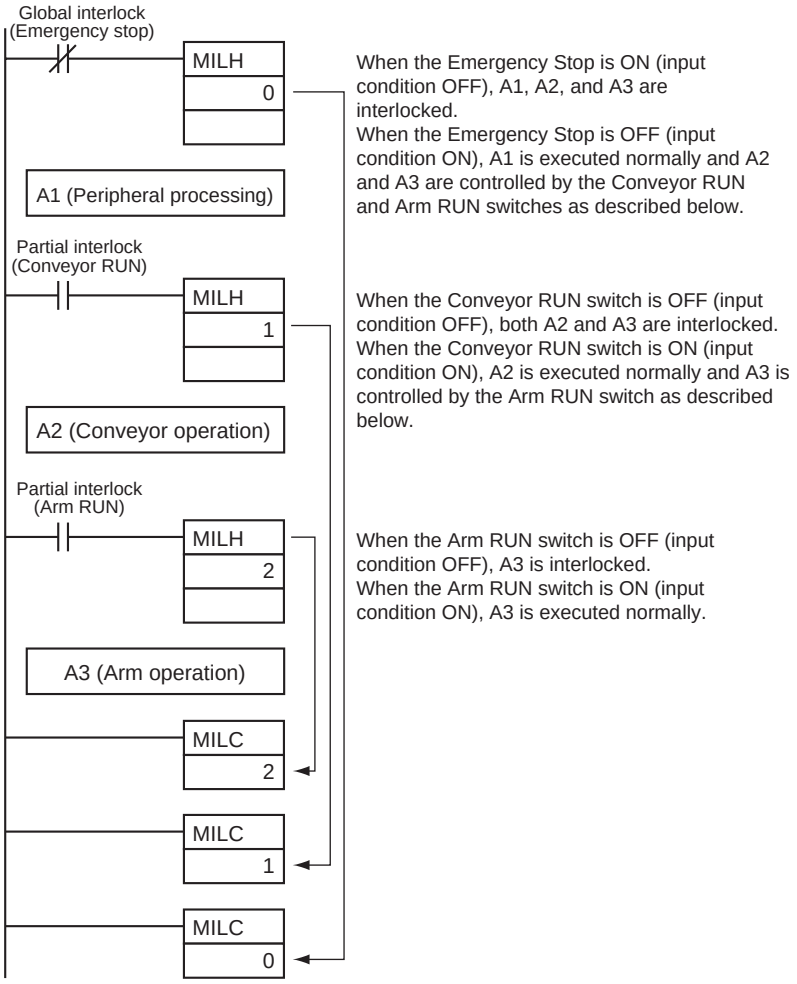
• Example 2

Interlocking the entire program with one condition and interlocking two overlapping parts of the program with other conditions (2 nesting levels)



- A1, A2, and A3 are interlocked when the Emergency Stop Button is ON.
- A2 and A3 are interlocked when Conveyor RUN is OFF.

- A3 is interlocked when Arm RUN is OFF.



Differences between MILH(517) and MILR(518)

Differentiated instructions (DIFU(013), DIFD(014), or instructions with a @ or % prefix) operate differently in interlocks created with MILH(517) and MILR(518).

When a program section is interlocked with MILR(518), a differentiated instruction **will not** be executed when the interlock is cleared even if the differentiation condition was activated during the interlock (comparing the status of the execution condition when the interlock started to its status when the interlock was cleared).

When a program section is interlocked with MILH(517), a differentiated instruction **will** be executed when the interlock is cleared if the differentiation condition was activated during the interlock (comparing the status of the execution condition when the interlock started to its status when the interlock was cleared).

Instruction	Operation of Differentiated Instructions
MILH(517) MULTI-INTERLOCK DIFFERENTIATION HOLD	A differentiated instruction (DIFU, DIFD, or instruction with a @ or % prefix) **will** be executed after the interlock is cleared if the differentiation condition of the instruction was established while the instruction was interlocked. (The status of the execution condition when the interlock started is compared to its status when the interlock was cleared.)
MILR(518) MULTI-INTERLOCK DIFFERENTIATION RELEASE	A differentiated instruction (DIFU, DIFD, or instruction with a @ or % prefix) **will not** be executed after the interlock is cleared even if the differentiation condition of the instruction was established while the instruction was interlocked.

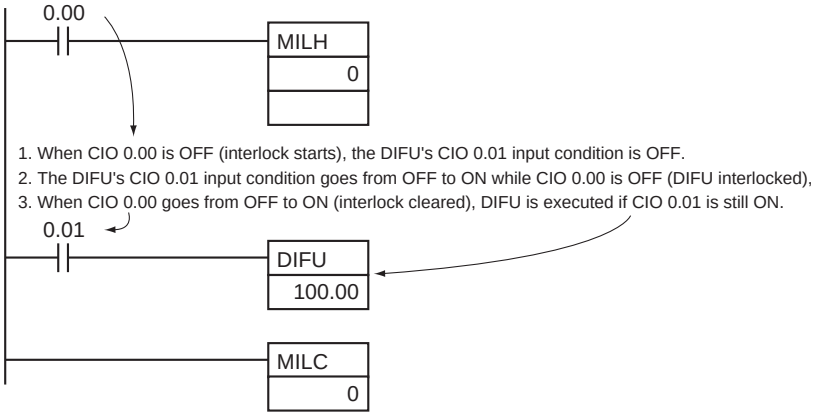
- Operation of Differentiated Instructions in an MILH(517) Interlock

  If there is a differentiated instruction (DIFU, DIFD, or instruction with a @ or % prefix) between MILH(517) and the corresponding MILC(519), that instruction **will** be executed after the interlock is cleared if the differentiation condition of the instruction was established. (The system compares the execution condition's status when the interlock started to its status when the interlock was cleared.)

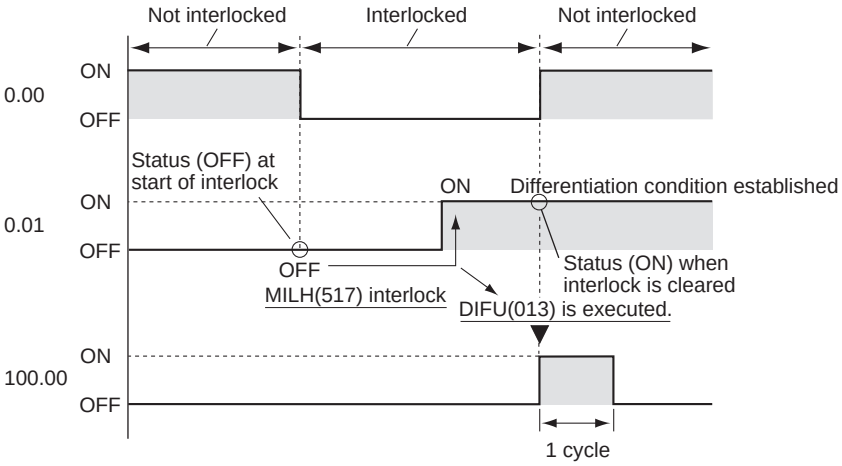
  In the same way, a differentiated instruction will be executed if its execution condition is established at the same time that the interlock is started or cleared.

  Many other conditions in the program may cause the differentiation condition to be reset even if it was established during the interlock. In this case, the differentiation instruction will not be executed when the interlock is cleared.

  - Example
    When a DIFFERENTIATE UP (DIFU(013)) instruction is being used and the input condition is OFF when the interlock starts and ON when the interlock is cleared, DIFU(013) **will** be executed when the interlock is cleared. (Differentiated instructions operate the same in the MILH(517) interlock as they would in an IL(002) interlock.)



Timing Chart



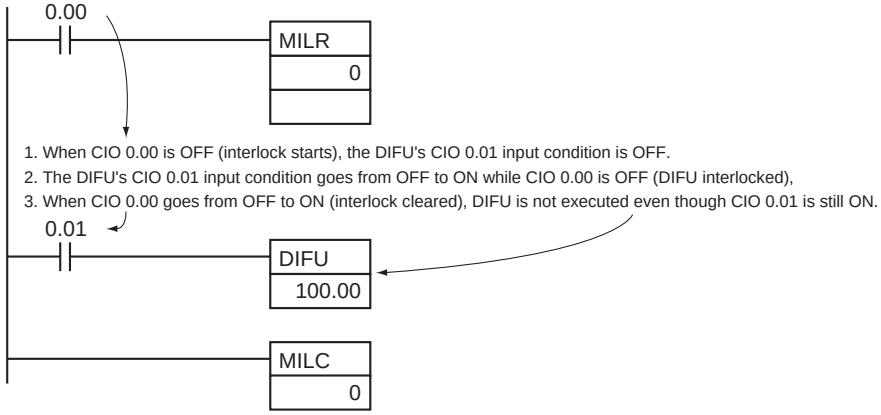
- Operation of Differentiated Instructions in an MILR(518) Interlock

  If there is a differentiated instruction (DIFU, DIFD, or instruction with a @ or % prefix) between MILR(518) and the corresponding MILC(519), that instruction **will not** be executed after the interlock is cleared even if the differentiation condition of the instruction was established. (The system compares the execution condition's status in the cycle when the interlock started to its status in the cycle when the interlock was cleared.)

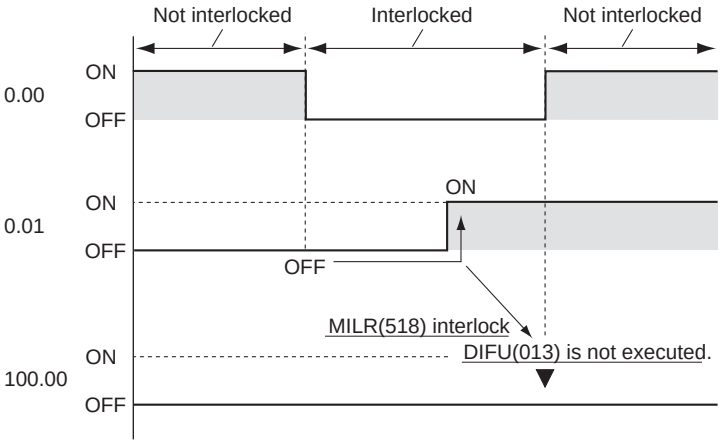
  In the same way, a differentiated instruction will not be executed if its execution condition is established at the same time that the interlock is started or cleared.

  - Example

    When a DIFFERENTIATE UP (DIFU(013)) instruction is being used and the input condition is OFF when the interlock starts and ON when the interlock is cleared, DIFU(013) **will not** be executed when the interlock is cleared.



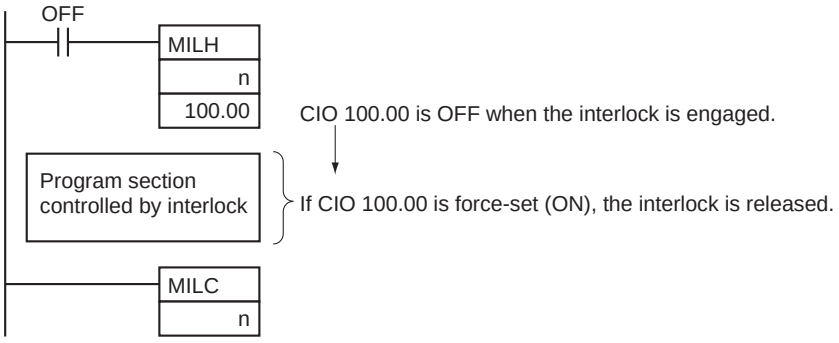
Timing Chart



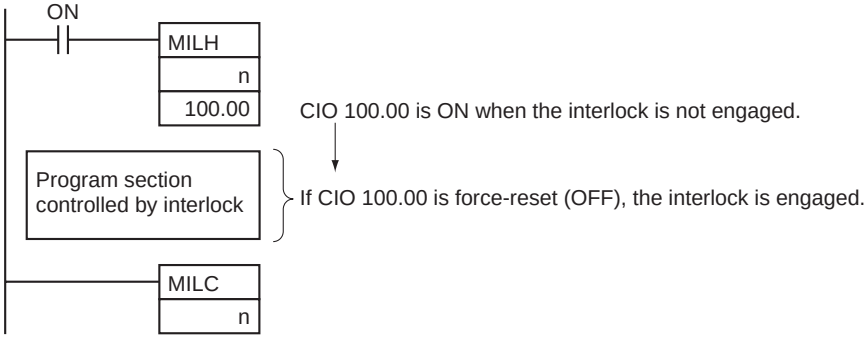
Controlling Interlock Status from the CX-Programmer

An interlock can be engaged or released manually by force-resetting or force-setting the Interlock Status Bit (specified with operand D of MILH(517) and MILR(518)) from the CX-Programmer. The forced status of the Interlock Status Bit has priority and overrides the interlock status calculated by program execution.

Force-set: Releases the interlock.

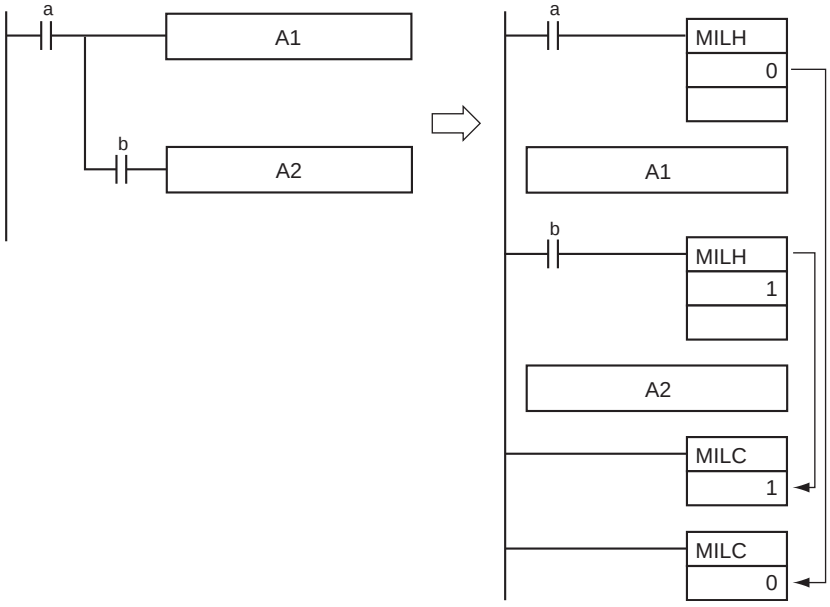


Force-reset: Engages the interlock.



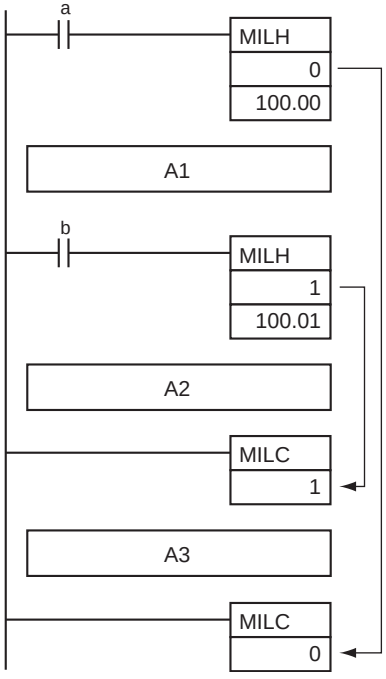
Note Program operation can be switched more efficiently by using interlocks with MILH(517) or MILR(518).

Instead of switching processing with compound conditions, insert an MILH(517) or MILR(518) instruction before each process and an MILC(519) instruction after each process.



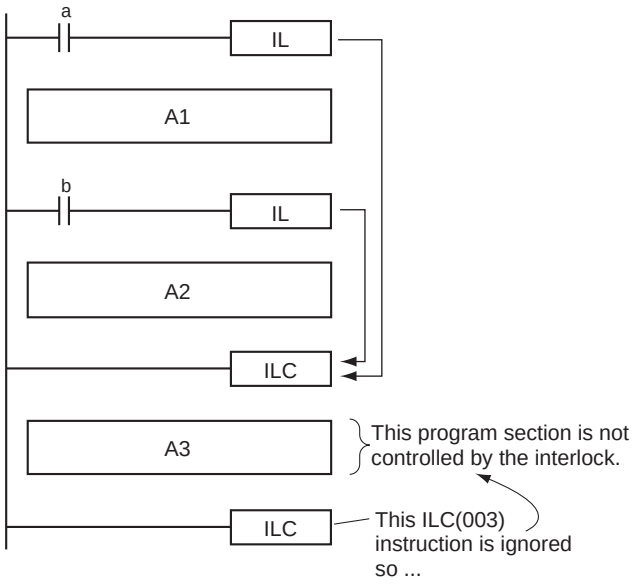
Unlike the IL(002) interlocks, MILH(517) and MILR(518) interlocks can be nested, so the operation of similar programs will be different if MILH(517) or MILR(518) is used instead of ILC(002).

Program with MILH(517)/MILC(519) Interlocks



Execution condition		Program section		
a	**b**	**A1**	**A2**	**A3**
OFF	ON	Interlocked	Interlocked	Not interlocked
	OFF			
ON	OFF	Not interlocked	Interlocked	Not interlocked
ON	ON	Not interlocked	Not interlocked	Not interlocked

Program with IL(002)/ILC(003) Interlocks



Execution condition		Program section		
a	**b**	**A1**	**A2**	**A3**
OFF	ON	Interlocked	Interlocked	Not interlocked (Not controlled by the IL(002)/ ILC(003) interlock.)
	OFF			
ON	OFF	Not interlocked	Interlocked	
ON	ON	Not interlocked	Not interlocked	

If there are bits which you want to remain ON in a program section interlocked by MILH(517) or MILR(518), set these bits to ON with SET just before the MILH(517) or MILR(518) instruction.

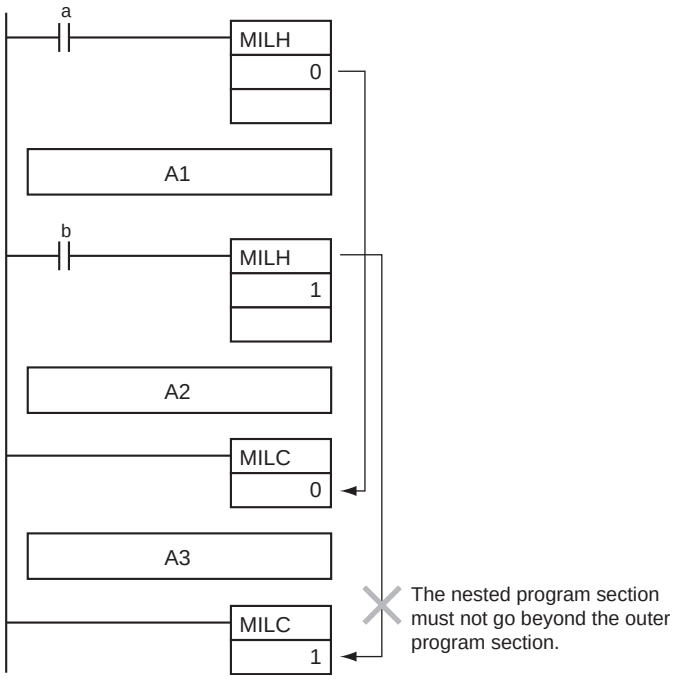
Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

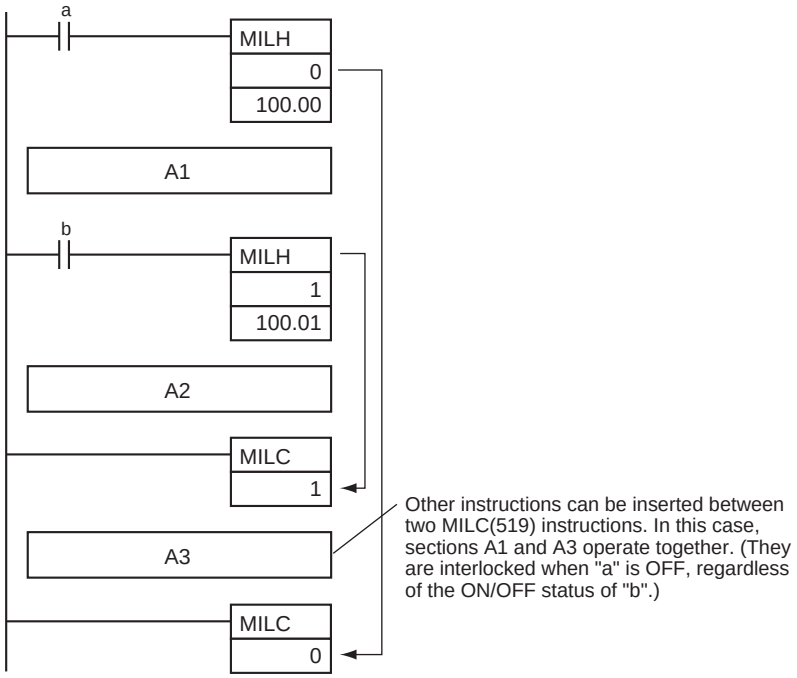
The cycle time is not shortened when a section of the program is interlocked by MILH(517) or MILR(518) because the interlocked instructions are executed internally.

When nesting interlocks, assign interlock numbers so that the nested program section does not exceed the outer program section.

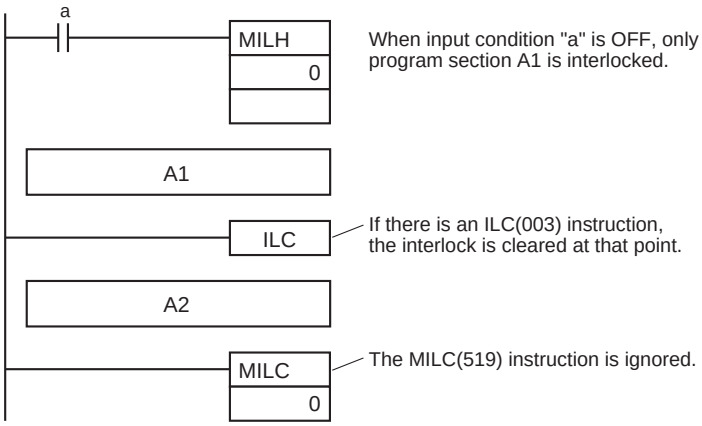


Execution condition		Program section		
a	**b**	**A1**	**A2**	**A3**
OFF	ON	Interlocked	Interlocked	Not interlocked
	OFF			
ON	OFF	Not interlocked	Interlocked	Interlocked
	ON	Not interlocked	Not interlocked	Not interlocked

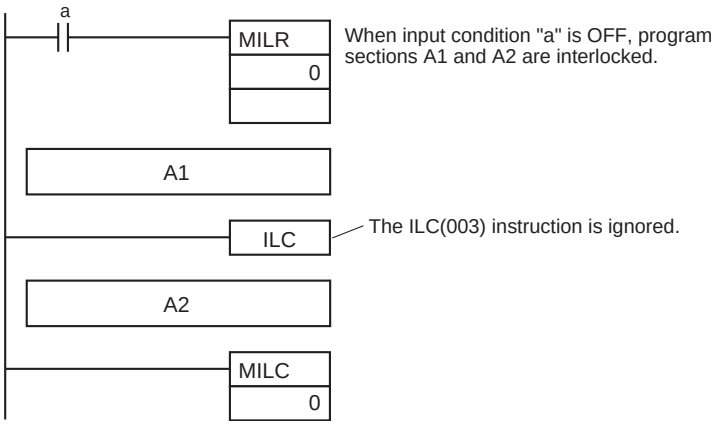
Other instructions can be input between the MILC(519) instructions, as shown in the following diagram.



If there is an ILC(003) instruction between an MILH(517) and MILC(519) pair, the program section between MILH(517) and ILC(003) will be interlocked.

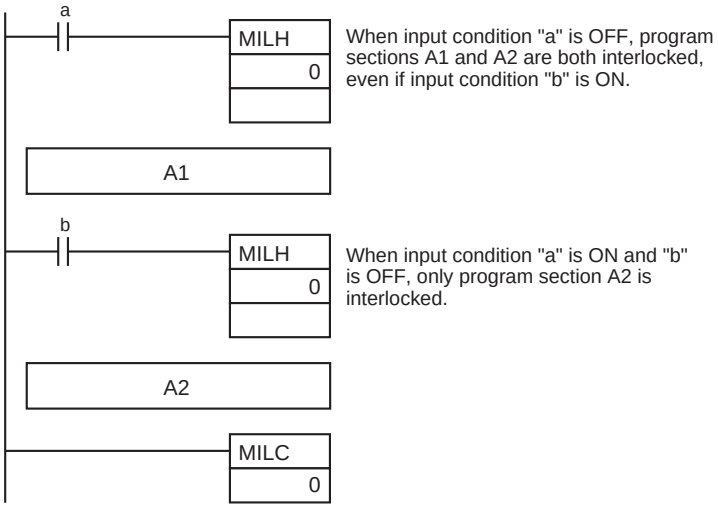


If there is an ILC(003) instruction between an MILR(518) and MILC(519) pair, the ILC(003) instruction will be ignored and the full program section between MILR(518) and MILC(519) will be interlocked.



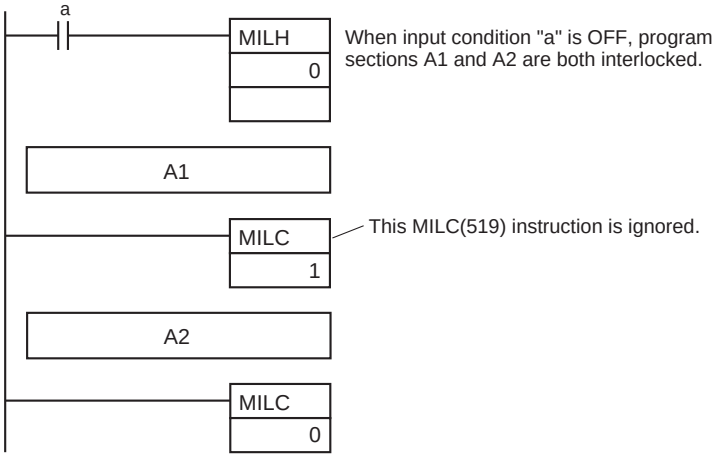
If there is another MILH(517) or MILR(518) instruction with the same interlock number between an MILH(517) and MILC(519) pair and the first MILH(517) instruction's interlock is engaged, the second MILH(517)/MILR(518) will not operate.

If there is another MILH(517) or MILR(518) instruction with the same interlock number between an MILH(517) and MILC(519) pair and the first MILH(517) instruction's interlock is not engaged, the second MILH(517)/MILR(518) will operate normally.



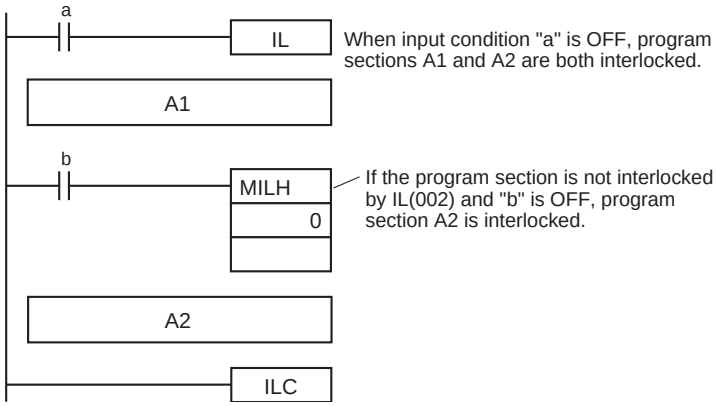
Note The MILR(518) interlocks operate in the same way if there is another MILH(517) or MILR(518) instruction with the same interlock number between an MILR(518) and MILC(519) pair.

If there is an MILC(519) instruction with a different interlock number between an MILH(517)/MILR(518) and MILC(519) pair, that MILC(519) instruction will be ignored.

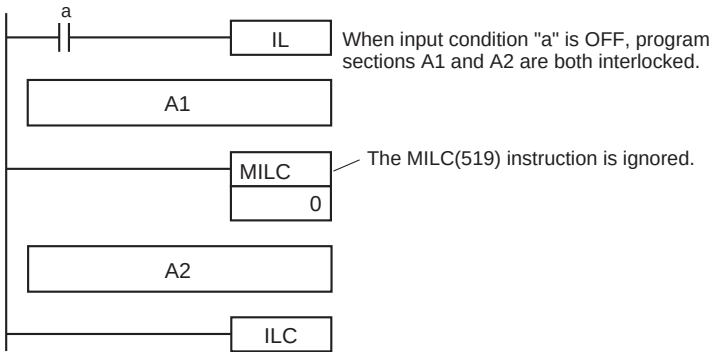


If there is an MILH(517) instruction between an IL(002) and ILC(003) pair and the IL(002) interlock is engaged, the MILH(517) instruction has no effect. In this case, the program section between IL(002) and ILC(003) will be interlocked.

If the IL(002) interlock is not engaged and the MILH(517) instruction's execution condition (b in this case) is OFF, the program section between MILH(517) and ILC(003) will be interlocked.



If there is an MILC(519) instruction between an IL(002) and ILC(003) pair, that MILC(519) instruction will be ignored and the entire program section between IL(002) and ILC(003) will be interlocked.

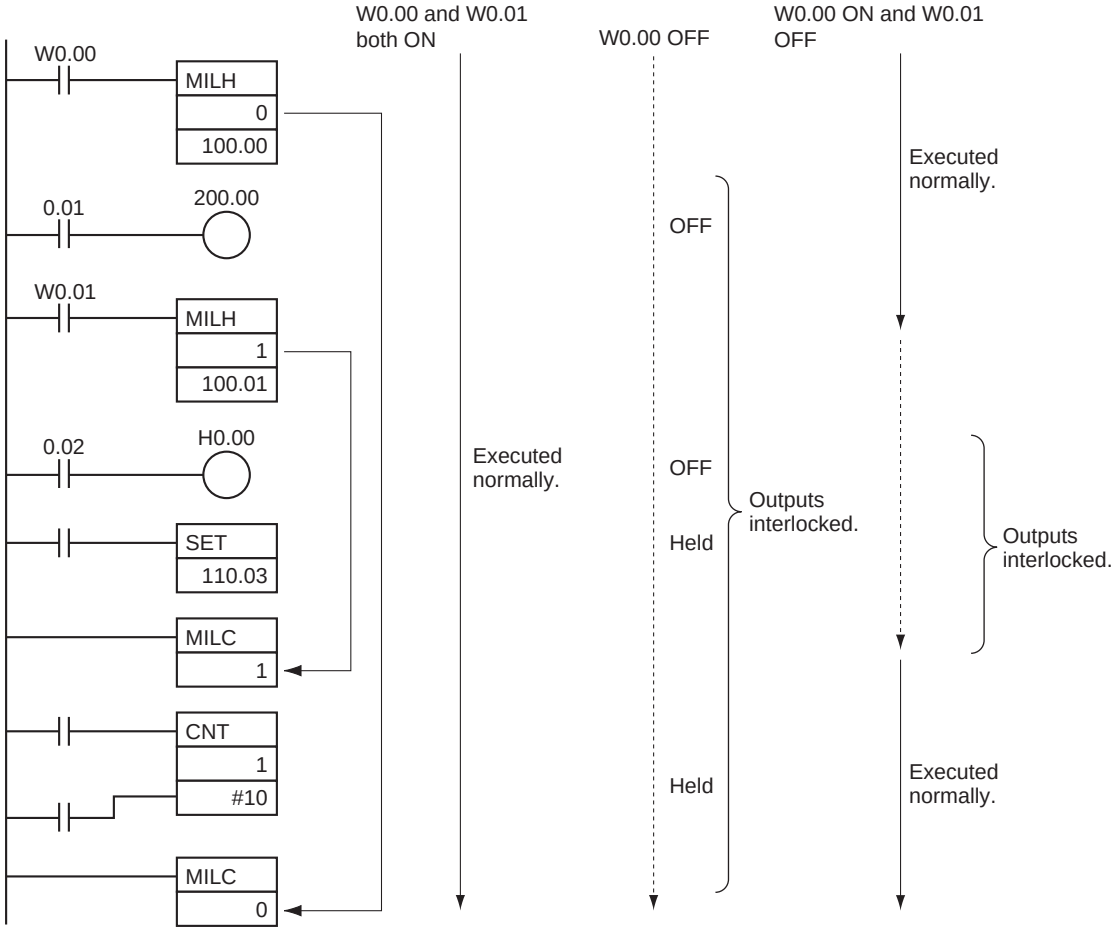


Examples

When W0.00 and W0.01 are both ON, the instructions between MILH(517) with interlock number 0 and MILC(519) with interlock number 0 are executed normally.

When W0.00 is OFF, the instructions between MILH(517) with interlock number 0 and MILC(519) with interlock number 0 are interlocked.

When W0.00 is ON and W0.01 are OFF, the instructions between MILH(517) with interlock number 1 and MILC(519) with interlock number 1 are interlocked. The other instructions are executed normally.

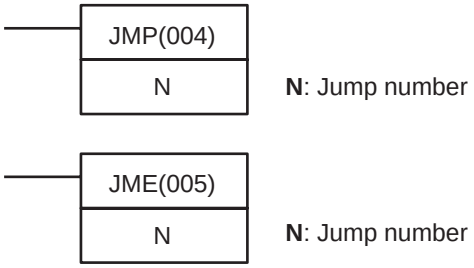


3-4-6 JUMP and JUMP END: JMP(004) and JME(005)

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



Variations

Variations	Jumps when OFF/Does Not Jump when ON	JMP(004)
Immediate Refreshing Specification		Not supported

Variations	Executed Each Cycle for ON Condition	JME(005)
Immediate Refreshing Specification		Not supported

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).

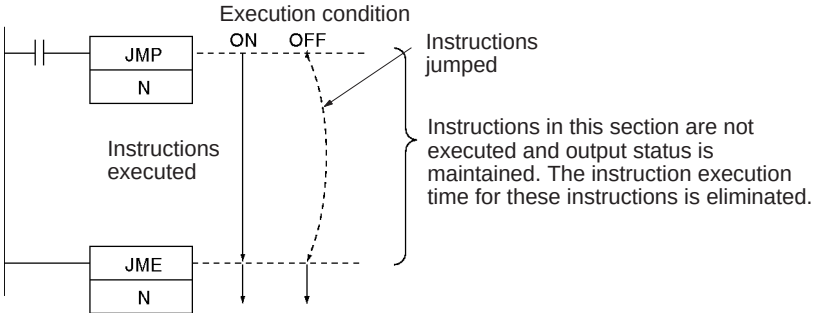
Operand Specifications

Area	N	
	JMP(004)	JME(005)
CIO Area	CIO 0 to CIO 6143	---
Work Area	W0 to W511	---
Holding Bit Area	H0 to H511	---
Auxiliary Bit Area	A0 to A959	---
Timer Area	T0000 to T4095	---
Counter Area	C0000 to C4095	---
DM Area	D0 to D32767	---
Indirect DM addresses in binary	@ D0 to @ D32767	---
Indirect DM addresses in BCD	*D0 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 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. In contrast, NOP(000) processing is performed for instructions between JMP0(515) and JME0(516), so the cycle time is not reduced as much with those jump instructions.

The following table compares the various jump instructions.

Item	JMP(004) JME(005)	CJP(510) JME(005)	CJPN(511) JME(005)	JMP0(515) JME0(516)
Execution condition for jump	OFF	ON	OFF	OFF
Number allowed	256 total			No limit
Instruction processing when jumped	Not executed.			NOP(000) processing
Instruction execution time when jumped	None			Same as NOP(000) instructions
Status of outputs (bits and words) when jumped	Bits and words maintain their previous status.			
Status of operating timers when jumped	Operating timers continue timing.			
Processing in block programs	Always jump.	Jump when ON.	Jump when OFF.	Not allowed.

Flags (JMP)

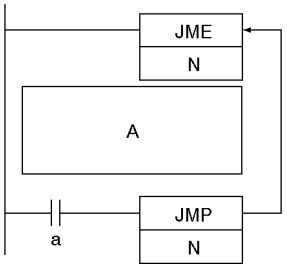
Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 255 (0000 to 00FF hex). ON if there is a JMP(004) in the program without a JME(005) with the same jump number. ON if there is a JMP(004) in the task without a JME(005) with the same jump number in the task. OFF in all other cases.

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 because the PVs are updated even when the timer instruction is not being executed.

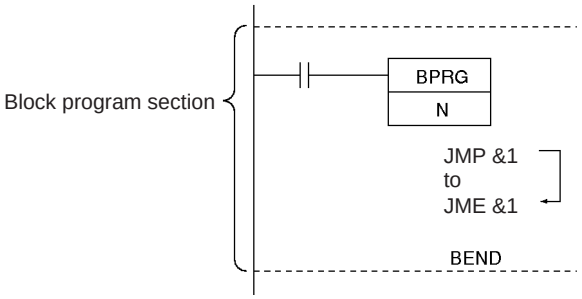
When there are two or more JME(005) instructions with the same jump number, only the instruction with the lower address will be valid. The JME(005) with the higher program address will be ignored.

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.



Program section A is executed repeatedly as long as execution condition a is OFF.

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.

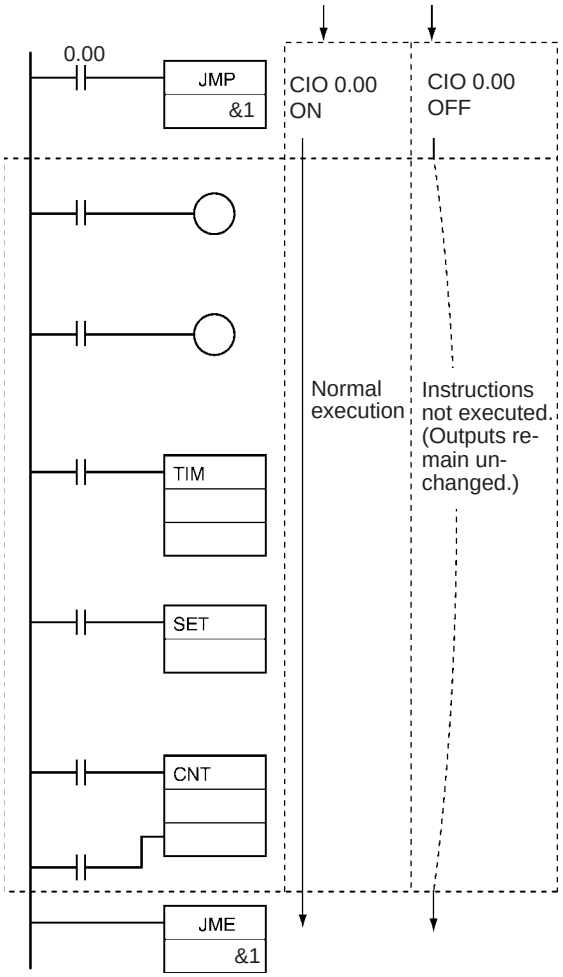
The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between JMP(004) and JME(005). When DIFU(013), DIFD(014), or a differentiated instruction is executed in an jumped section immediately after 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).

Examples

Basic Operation

When CIO 0.00 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 0.00 is ON in the following example, the instructions between JMP(004) and JME(005) are executed normally.



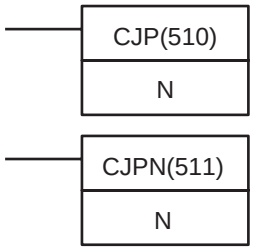
3-4-7 CONDITIONAL JUMP: CJP(510)/CJPN(511)

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)
Immediate Refreshing Specification		Not supported

Variations	Jumps when OFF/Does Not Jump when ON	CJPN(511)
Immediate Refreshing Specification		Not supported

Variations	Executed Each Cycle for ON Condition	JME(005)
Immediate Refreshing Specification		Not supported

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 0 to CIO 6143		---
Work Area	W0 to W511		---
Holding Bit Area	H0 to H511		---
Auxiliary Bit Area	A0 to A959		---
Timer Area	T0000 to T4095		---
Counter Area	C0000 to C4095		---
DM Area	D0 to D32767		---
Indirect DM addresses in binary	@ D0 to @ D32767		---
Indirect DM addresses in BCD	*D0 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 IR15		---

Description

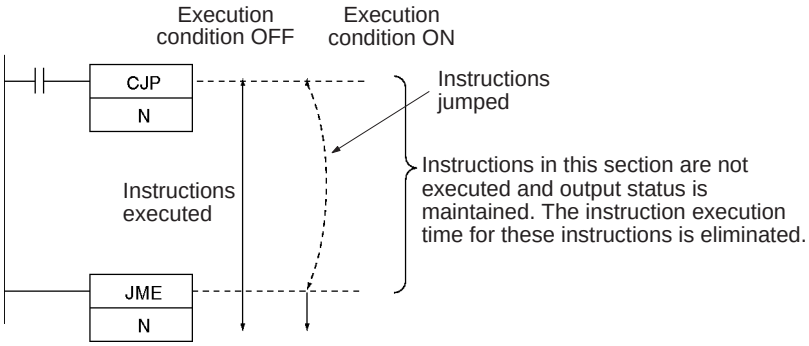
The operation of CJP(510) and CJPN(511) differs only in the execution condition. CJP(510) jumps to the first JME(005) when the execution condition is ON and CJPN(511) jumps to the first JME(005) when the execution condition is OFF.

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.

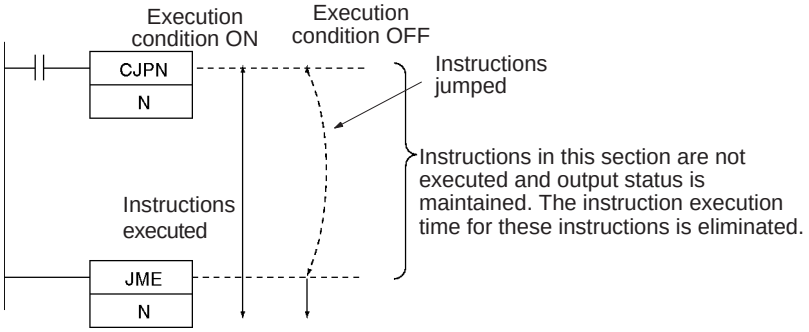
When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number.



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). ON if N is not within the specified range of 0 to 255 (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.

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.

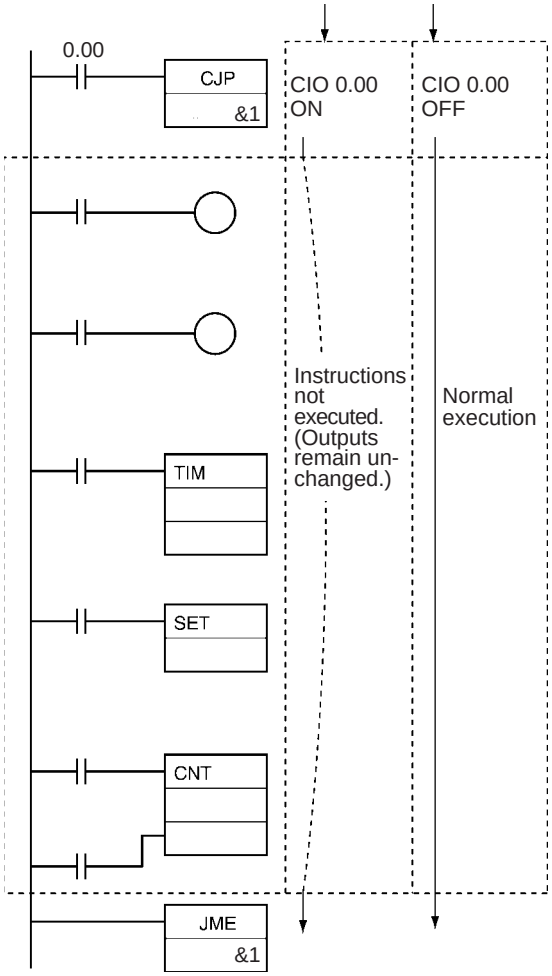
When a CJP(510) or CJPN(511) instruction is programmed in a task, there must be a JME(005) with the same jump number because jumps between tasks are not allowed. An error will occur if a corresponding JME(005) instruction is not programmed in the same task.

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 0.00 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 0.00 is OFF in the following example, the instructions between CJP(510) and JME(005) are executed normally.



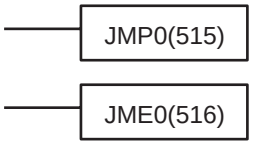
Note For CJPN(511), the ON/OFF status of CIO 0.00 would be reversed.

3-4-8 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)
Immediate Refreshing Specification		Not supported

Variations	Executed Each Cycle for ON Condition	JME0(516)
Immediate Refreshing Specification		Not supported

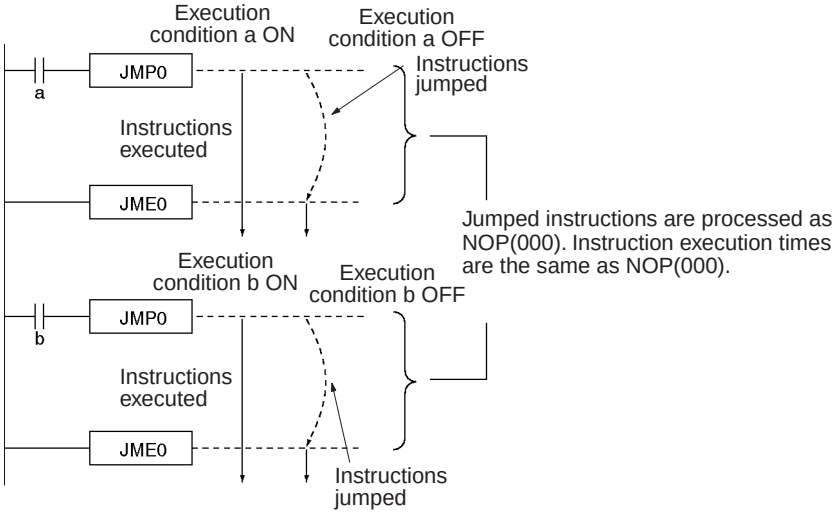
Applicable Program Areas

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.

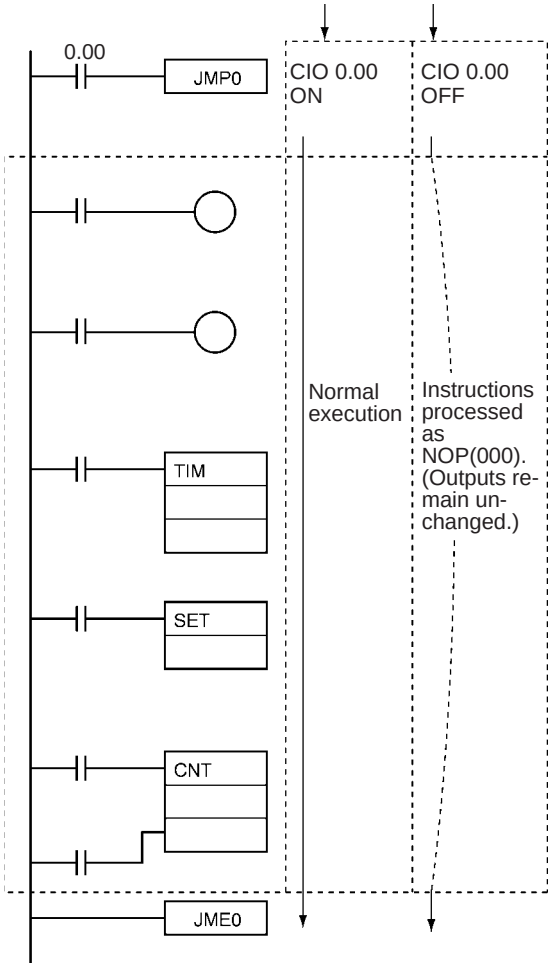
JMP0(515) and JME0(516) cannot be used in block programs.

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 0.00 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 0.00 is ON in the following example, the instructions between JMP0(515) and JME0(516) are executed normally.

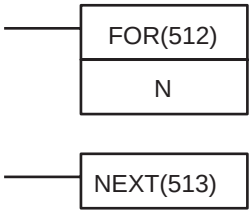


3-4-9 FOR-NEXT LOOPS: FOR(512)/NEXT(513)

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)
Immediate Refreshing Specification		Not supported

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

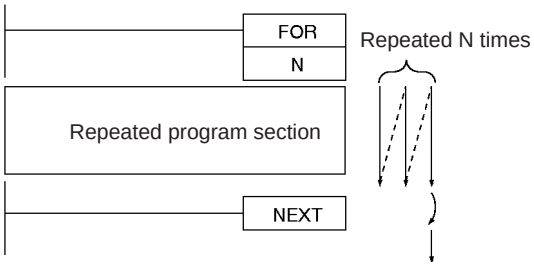
Area	N
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A0 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	#0000 to #FFFF (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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15

Description

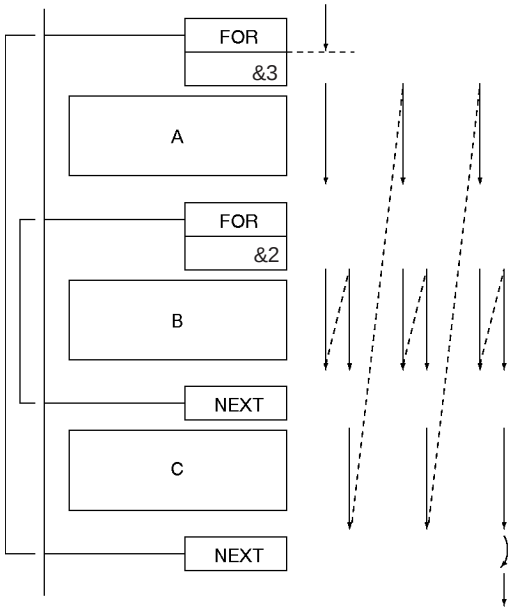
The instructions between FOR(512) and NEXT(513) are executed N times and then program execution continues with the instruction after NEXT(513). The BREAK(514) instruction can be used to cancel the loop.

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

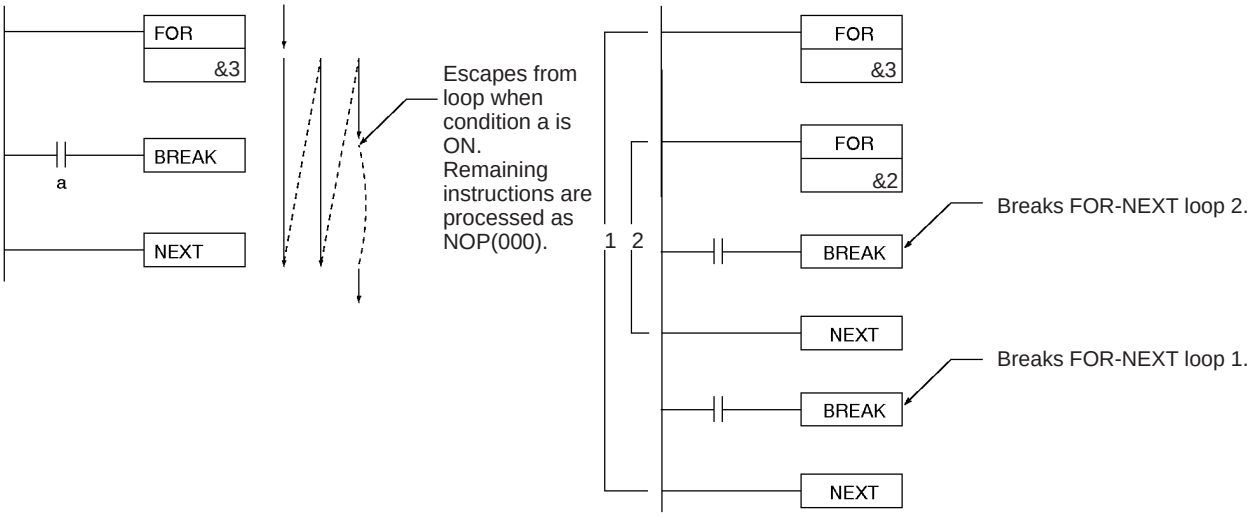
Loops can be used to process tables of data with a minimum amount of programming.



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.

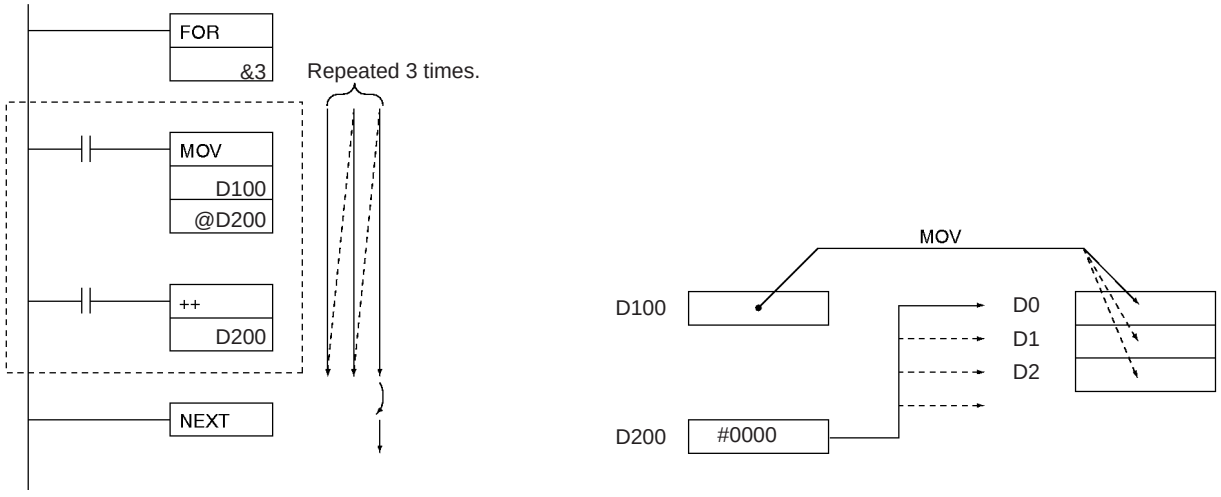
The following instructions cannot be used within FOR-NEXT loops:

- 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 D100 to the address indicated in D200 and then increments the content of D200 by 1.



3-4-10 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

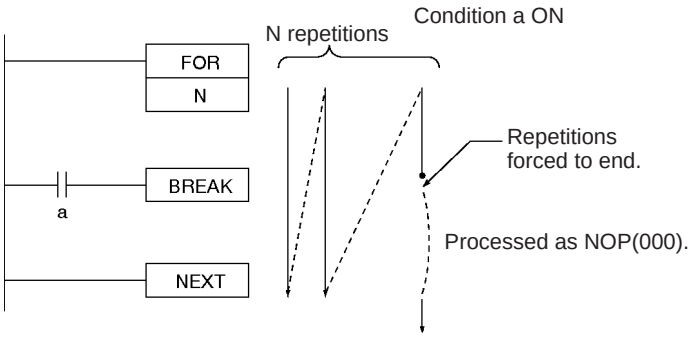
Variations	**Executed Each Cycle for ON Condition**	BREAK(514)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

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/TIMX	---/550	171
HIGH-SPEED TIMER	TIMH/TIMHX	015/551	175
ONE-MS TIMER	TMHH/TIMHHX	540/552	179
ACCUMULATIVE TIMER	TTIM/TTIMX	087/555	182
LONG TIMER	TIML/TIMLX	542/553	185
MULTI-OUTPUT TIMER	MTIM/MTIMX	543/554	188
COUNTER	CNT/CNTX	---/546	194
REVERSIBLE COUNTER	CNTR/CNTRX	012/548	197
RESET TIMER/COUNTER	CNR/CNRX	545/547	201

Refresh Methods for Timer/Counter PV

■ Overview

The refresh method for present values timer and counter instructions can be set to either BCD or binary for CP-series CPU Units.

Using binary data instead of BCD allows the SV range for timers and counter to be increased from 0 to 9999 to 0 to 65535. It also enables using binary data calculated with other instructions directly as a timer/counter SV. The refresh method is valid even when setting an SV indirectly (i.e., using the contents of memory word). (That is, the contents of the addressed word is taken as either BCD or binary data according to the refresh method that is set.)

Refer to the *CP Series CP1H Operation Manual* for details on refresh methods.

■ Applicable Instructions

Classification	Instruction	Mnemonic	
		BCD	Binary
Timer/counter instructions	TIMER	TIM	TIMX(550)
	HIGH-SPEED TIMER	TIMH(015)	TIMHX(551)
	ONE-MS TIMER	TMHH(540)	TMHHX(552)
	ACCUMULATIVE TIMER	TTIM(087)	TTIMX(555)
	LONG TIMER	TIML(542)	TIMLX(553)
	MULTI-OUTPUT TIMER	MTIM(543)	MTIMX(554)
	COUNTER	CNT	CNTX(546)
	REVERSIBLE COUNTER	CNTR(012)	CNTRX(548)
	RESET TIMER/COUNTER	CNR(545)	CNRX(547)
Block programming instructions	TIMER WAIT	TIMW(813)	TIMWX(816)
	HIGH-SPEED TIMER WAIT	TMHW(815)	TMHWX(817)
	COUNTER WAIT	CNTW(814)	CNTWX(818)

Basic Timer Specifications

The following table shows the basic specifications of the timers.

Item		TIM/TIMX(550)	TIMH(015)/ TIMHX(551)	TMHH(540)/ TMHHX(552)	TTIM(087)/ TTIMX(555)	TIML(542)/ TIMLX(553)	MTIM(543)/ MTIMX(554)
Timing method		Decrementing	Decrementing	Decrementing	Incrementing	Decrementing	Incrementing
Timing units		0.1 s	0.01 s	0.001 s	0.1 s	0.1 s	0.1 s
Max. SV		TIM: 999.9 s TIMX: 6,553.5 s	TIMH: 99.99 s TIMHX: 655.35 s	TMHH: 9.999 s TMHHX: 65.535 s	TTIM: 999.9 s TTIMX: 6,553.5 s	TIML: 115 days TIMLX: 49,710 days	MTIM: 999.9 s MTIMX: 6,553.5 s
Outputs/ instruction		1	1	1	1	1	8
Timer numbers		Used	Used	Used	Used	Not used	Not used
Comp. flag refreshing		At execution	At execution	By interrupt every 1 ms	At execution	At execution	At execution
Timer PV refreshing		See note 1.	See note 2.	Every 1 ms	At execution	At execution	At execution
Value after reset	Comp. flags	OFF	OFF	OFF	OFF	OFF	OFF
	PVs	SV	SV	SV	0	SV	0

Note

(1) TIM PVs are refreshed at execution for all times and also every 100 ms for T0000 to T0015.

(2) TIMH(015)/TIMHX(551) PVs are refreshed at execution for all times and also every 10 ms for T0000 to T0015.

Timer Operation

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

Item		TIM/ TIMX(550)	TIMH(015)/ TIMHX(551)	TMHH(540)/ TMHHX(552)	TTIM(087)/ TTIMX(555)	TIML(542)/ TIMLX(553)	MTIM(543)/ MTIMX(554)
Operating mode change		PV = 0 Completion Flag = OFF				---	---
Power interrupt/reset		PV = 0 Completion Flag = OFF				---	---
Execution of CNR(545)/ CNRX(547)		Binary: PV = FFFF, Completion Flag = OFF BCD: PV = FFFF or 9999, Completion Flag = OFF				Not applicable	Not applicable
Operation in jumped program section (JMP(004)-JME(005))		Operating timers continue timing.			Timer status is maintained.		
Operation in interlocked program section (IL(002)-ILC(003))		PV = SV Completion Flag = OFF			Timer status maintained.	PV = SV Comp. flag = OFF	Timer status maintained.
Forced set	Comp. flags	ON				---	---
	PVs	Set to 0.				---	---
Forced reset	Comp. flags	OFF				---	---
	PVs	Reset to SV.			Set to 0.	---	---

3-5-1 TIMER: TIM/TIMX(550)

Purpose

TIM or TIMX(550) operates a decrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 999.9 s for TIM and 0 to 6,553.5 s for TIMX(550). The timer accuracy is 0 to 0.01 s.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	TIM N S **N**: Timer number **S**: Set value	N: 0000 to 4095 (decimal) S: #0000 to #9999 (BCD)
Binary	TIMX(550) N S **N**: Timer number **S**: Set value	N: 00000 to 4095 (decimal) S: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Variations

Variations		TIM/TIMX(550)
	Executed Each Cycle for ON Condition	TIM/TIMX(550)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 4095 (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/TIMX(550) is executed.)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767

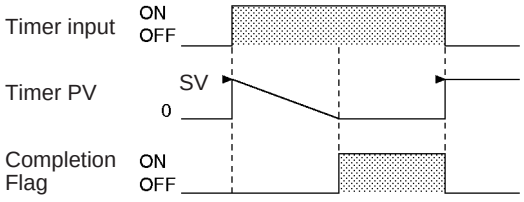
Area	N	S
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

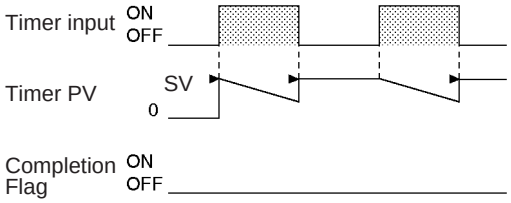
When the timer input is OFF, the timer specified by N is reset, i.e., the timer’s PV is reset to the SV and its Completion Flag is turned OFF.

When the timer input goes from OFF to ON, TIM/TIMX(550) 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 in BCD mode and S does not contain BCD data. OFF in all other cases.
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

Precautions

Timer numbers are shared by the TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TTIM(087), TTIMX(555), TIMW(813), TIMWX(816), TMHW(815), and TMHWX(817) instructions. If two timers share the same timer number, but are not used simultaneously, a duplication error will be generated when the program is checked, but the timers will operate normally. Timers which share the same timer number will not operate properly if they are used simultaneously.

Timers created with timer numbers 16 to 4095 will not operate properly when the CPU Unit cycle time exceeds 100 ms. Use timer numbers 0 to 15 when the cycle time is longer than 100 ms.

The present value of timers programmed with timer numbers 0 to 15 will be updated even when the timer is on standby. The present value of timers programmed with timer numbers 16 to 4095 will be held when the timer is on standby.

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.[1]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	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

(1) If the IOM Hold Bit (A500.12) has been turned ON, the status of timer Completion Flags and PVs will be maintained when the operating mode is changed.

(2) If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.

(3) The PV will be set to the SV when TIM/TIMX(550) is executed.

When TIM/TIMX(550) 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/TIMX(550) timer created with a timer number between 0 and 15 is in a jumped program section (JMP(004), CJMP(510), CJPN(511), JME(005)), the timer's PV will continue timing. The jumped TIM/TIMX(550) instruction will not be executed, but the PV will be refreshed each cycle after all tasks have been executed.

When a TIM/TIMX(550) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0. When a TIM/TIMX(550) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.

The operation of the = Flag and N Flag depends on the model of the CPU Unit. Refer to *Flags*, above, for details.

The timer's Completion Flag is refreshed only when TIM/TIMX(550) 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/TIMX(550) ↔ TIMH(015)/TIMHX(551) or TIM/TIMX(550) ↔ TMHH(540)/TMHHX(552)), be sure to reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

A TIM/TIMX(550) instruction's PV and Completion Flag can be refreshed in the following ways depending on the timer number that is used.

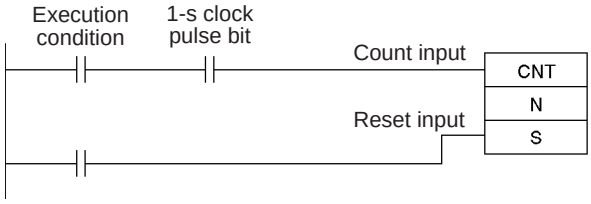
Timers Created with Timer Numbers 0000 to 2047

Execution of TIM/ TIMX(550)	The PV is updated every time that TIM/TIMX(550) 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.
100-ms interval refreshing	If the cycle time exceeds 100 ms, the timer's PV is updated every 100 ms.

Timers Created with Timer Numbers T0016 to T4095

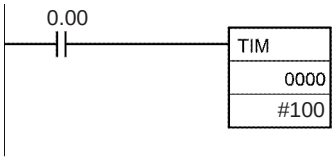
Execution of TIM	The PV is updated every time that TIM is executed. The Completion Flag is turned ON if the PV is 0. The Completion Flag is turned OFF if the PV is not 0.

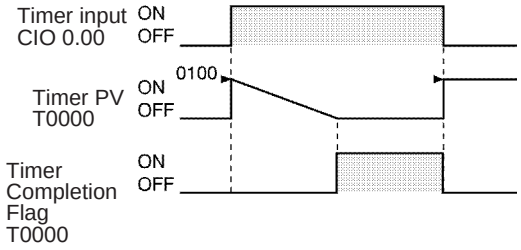
Timers are reset (PV = SV, Completion Flag OFF) by power interruptions unless the IOM Hold Bit (A500.12) is ON and the bit is protected in the PLC Setup. It is also possible use a clock pulse bit and a counter instruction to program a timer that will retain its PV in the event of a power interruption, as shown in the following diagram.



Example

When timer input CIO 0.00 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 0.00 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)/TIMHX(551)

Purpose

TIMH(015)/TIMHX(551) 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) and 0 to 655.35 s for TIMHX(551). The timer accuracy is 0 to 0.01 s.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	TIMH(015) N — **N**: Timer number S — **S**: Set value	N: 0000 to 4095 (decimal) S: #0000 to #9999 (BCD)
Binary	TIMHX(551) N — **N**: Timer number S — **S**: Set value	N: 00000 to 4095 (decimal) S: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Variations

Variations		
	Executed Each Cycle for ON Condition	TIMH(015)/ TIMHX(551)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 4095 (decimal).

S: Set Value

The set value must be between #0000 and 9999 in BCD mode.

Operand Specifications

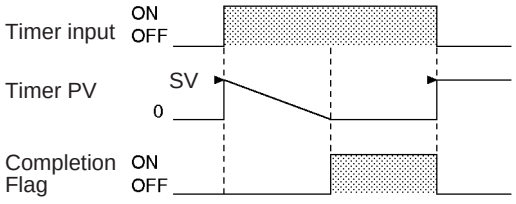
Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
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	

Description

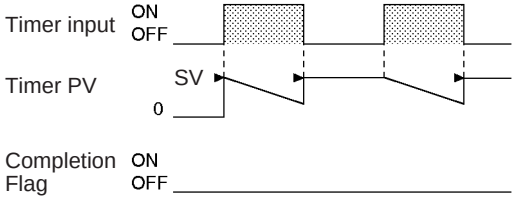
When the timer input is OFF, the timer specified by N is reset, i.e., the timer’s PV is reset to the SV and its Completion Flag is turned OFF.

When the timer input goes from OFF to ON, TIMH(015)/TIMHX(551) 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 in BCD mode and S does not contain BCD data. OFF in all other cases.
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

Precautions

Timer numbers are shared by the TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TTIM(087), TTIMX(555), TIMW(813), TIMWX(816), TMHW(815), and TMHWX(817) instructions. If two timers share the same timer number, but are not used simultaneously, a duplication error will be generated when the program is checked, but the timers will operate normally. Timers which share the same timer number will not operate properly if they are used simultaneously.

Timers created with timer numbers 16 to 4095 will not operate properly when the CPU Unit cycle time exceeds 100 ms. Use timer numbers 0 to 15 when the cycle time is longer than 100 ms.

TIMH(015)/TIMHX(551) timers created with timer numbers 0 to 15 are refreshed every 10 ms. Use these timer numbers when the PV is being referenced in the user program.

The present value of timers programmed with timer numbers 0 to 15 will be updated even when the timer is on standby. The present value of timers programmed with timer numbers 16 to 4095 will be held when the timer is on standby.

The operation of the = Flag and N Flag depends on the model of the CPU Unit. Refer to *Flags*, above, for details.

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.[1]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	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

(1) If the IOM Hold Bit (A500.12) has been turned ON, the status of timer Completion Flags and PVs will be maintained when the operating mode is changed.

(2) If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.

(3) The PV will be set to the SV when TIMH(015)/TIMHX(551) is executed.

When an operating TIMH(015)/TIMHX(551) timer created with a timer number between 0 and 15 is in a jumped program section (JMP(004), CJMP(510), CJPN(511), JME(005)), the timer's PV will continue timing. (The jumped TIMH(015)/TIMHX(551) 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)/TIMHX(551) 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)/TIMHX(551) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0. When a TIMH(015)/TIMHX(551) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.

The operation of the = Flag and N Flag depends or the model of CPU Unit. Refer to *Flags* for details.

The timer's Completion Flag is refreshed only when TIMH(015)/TIMHX(551) 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)/TIMHX(551) ↔ TIM/TIMX(550) or TIMH(015)/TIMHX(551) ↔ TMHH(540)/TMHHX(552)), be sure to reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

A TIMH(015)/TIMHX(551) instruction's PV and Completion Flag can be refreshed in the following ways depending on the timer number that is used.

Timers Created with Timer Numbers T0000 to T0015

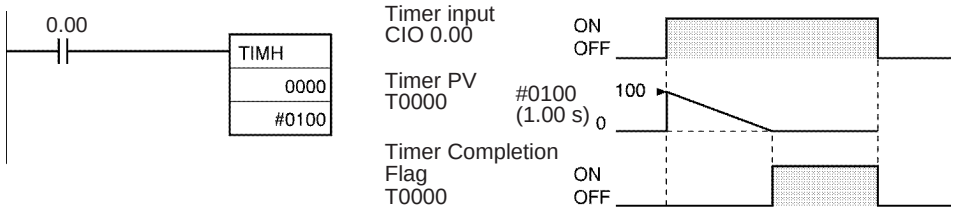
Execution of TIMH(015)/ TIMHX(551)	The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.
10-ms interval refreshing	The timer's PV is updated every 10 ms.

Timers Created with Timer Numbers T0016 to T4095

Execution of TIMH(015)/ TIMHX(551)	The PV is updated every time that TIMH(015) is executed. The Completion Flag is turned ON if the PV is 0. The Completion Flag is turned OFF if the PV is not 0.

Example

When timer input CIO 0.00 goes from OFF to ON in the following example, the timer PV will begin counting down from the SV (#0064 = 100 = 1.00 s). The Timer Completion Flag, T0000, will be turned ON when the PV reaches 0000. When CIO 0.00 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)/TMHHX(552)

Purpose

TMHH(540)/TMHHX(552) 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) and 0 to 65.535 for TMHHX(552). The timer accuracy is –0.001 to 0 s.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	TMHH(540) N — **N**: Timer number S — **S**: Set value	N: 0000 to 15 (decimal) S: #0000 to #9999 (BCD)
Binary	TMHHX(552) N — **N**: Timer number S — **S**: Set value	N: 00000 to 15 (decimal) S: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Variations

Variations		
	Executed Each Cycle for ON Condition	TMHH(540)/ TMHHX(552)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0015 (decimal).

S: Set Value

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

Operand Specifications

Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	0000 to 0015 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) "&" cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
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	

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)/TMHHX(552) 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 in BCD mode and S does not contain BCD data. OFF in all other cases.
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

Precautions

Timer numbers are shared by the TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TTIM(087), TTIMX(555), TIMW(813), TIMWX(816), TMHW(815), and TMHWX(817) instructions. If two timers share the same timer number, but are not used simultaneously, a duplication error will be generated when the program is checked, but the timers will operate normally. Timers which share the same timer number will not operate properly if they are used simultaneously.

The Completion Flag is updated only when TMHH(540)/TMHHX(552) is executed. The Completion Flag can thus be delayed by up to one cycle time from the actual set value.

The present value of timers programmed with timer numbers 0000 to 2047 will be updated even when the timer is on standby. The present value of timers programmed with timer numbers 2048 to 4095 will be held when the timer is on standby.

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.[1]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	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

(1) If the IOM Hold Bit (A500.12) has been turned ON, the status of timer Completion Flags and PVs will be maintained when the operating mode is changed.

(2) If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.

(3) The PV will be set to the SV when TMHH(540)/TMHHX(552) is executed.

When an operating TMHH(540)/TMHHX(552) timer is in a jumped program section (JMP(004), CJMP(510), CJPN(511), JME(005)), the timer's PV will continue timing. (The jumped TMHH(540)/TMHHX(552) instruction will not be executed, but the PV will be refreshed every 1 ms.)

When TMHH(540)/TMHHX(552) 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)/TMHHX(552) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0000. When a TMHH(540)/TMHHX(552) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.

The operation of the = Flag and N Flag depends on the model of the CPU Unit. Refer to *Flags*, above, for details.

If online editing is used to convert a timer to another kind of timer with the same timer number (such as TMHH(540)/TMHHX(552) ↔ TIM/TIMX(550) or TMHH(540)/TMHHX(552) ↔ TIMH(015)/TIMHX(551)), be sure to reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

A TMHH(540)/TMHHX(552) instruction's PV and Completion Flag are refreshed as shown in the following table.

Execution of TMHH(540)/ TMHHX(552)	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 ACCUMULATIVE TIMER: TTIM(087)/TTIMX(555)

Purpose

TTIM(087)/TTIMX(555) operates an incrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 999.9 s for TTIM(087) and 0 to 6,553.5 s for TTIMX(555). The timer accuracy is –0.01 to 0 s.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	Timer input — TTIM(087) / N — **N**: Timer number / S — **S**: Set value / Reset input	N: 0000 to 15 (decimal) S: #0000 to #9999 (BCD)
Binary	Timer input — TTIMX(555) / N — **N**: Timer number / S — **S**: Set value / Reset input	N: 00000 to 15 (decimal) S: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Variations

Variations		
	Executed Each Cycle for ON Condition	TTIM(087)/ TTIMX(555)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 to 4095 (decimal).

S: Set Value

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

Operand Specifications

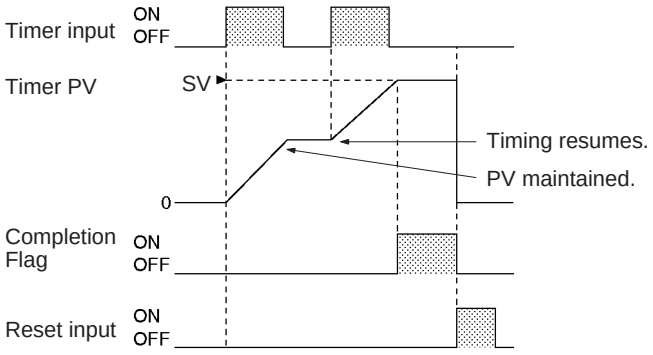
Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095

Area	N	S
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
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	

Description

When the timer input is ON, TTIM(087)/TTIMX(555) increments the PV. When the timer input goes OFF, the timer will stop incrementing the PV, but the PV will retain its value. The PV will resume timing when the timer input goes ON again. The timer’s Completion Flag will be turned ON when the PV reaches the SV.

The status of the timer’s PV and Completion Flag will be maintained after the timer times out. There are three ways to restart the timer: the timer’s PV can be changed to a non-zero value (by MOV(021), for example), the reset input can be turned ON, or CNR(545)/CNRX(547) can be executed.



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 in BCD mode and S does not contain BCD data. OFF in all other cases.

Precautions

Timer numbers are shared by the TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TTIM(087), TTIMX(555), TIMW(813), TIMWX(816), TMHW(815), and TMHWX(817) instructions. If two timers share the same timer number, but are not used simultaneously, a duplication error will be generated when the program is checked, but the timers will operate normally. Timers which share the same timer number will not operate properly if they are used simultaneously.

Timers will be reset or paused in the following cases. (When a TTIM(087)/TTIMX(555) timer is reset, its PV is reset to 0000 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.[1]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Retains previous status.	Retains previous status.
Operation in jumped program section (JMP(004)–JME(005))	Retains previous status.	Retains previous status.

Note

(1) If the IOM Hold Bit (A500.12) has been turned ON, the status of timer Completion Flags and PVs will be maintained when the operating mode is changed.

(2) If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.

(3) The PV will be set to the SV when TTIM(087)/TTIMX(555) is executed.

When TTIM(087)/TTIMX(555) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will retain its previous value (it will not be reset). Be sure to take this fact into account when TTIM(087)/TTIMX(555) is programmed between IL(002) and ILC(003).

When an operating TTIM(087)/TTIMX(555) timer is in a program section between JMP(004) and JME(005) and the program section is jumped, the PV will retain its previous value. Be sure to take this fact into account when TTIM(087)/TTIMX(555) is programmed between JMP(004) and JME(005).

When a TTIM(087)/TTIMX(555) timer is forced set, its Completion Flag will be turned ON and its PV will be reset to 0000. When a TTIM(087)/TTIMX(555) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to 0000. The forced set and forced reset operations take priority over the status of the timer and reset inputs.

The timer's PV is refreshed only when TTIM(087)/TTIMX(555) is executed, so the timer will not operate properly when the cycle time exceeds 100 ms because the timer increments in 100-ms units.

The timer's Completion Flag is refreshed only when TTIM(087)/TTIMX(555) 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.

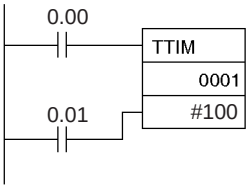
Typical timers such as TIM/TIMX(550) are decrementing counters and the PV shows the time remaining until the timer times out. The PV of TTIM(087)/TTIMX(555) shows how much time has elapsed, so the PV can be used unchanged in many calculations and display outputs.

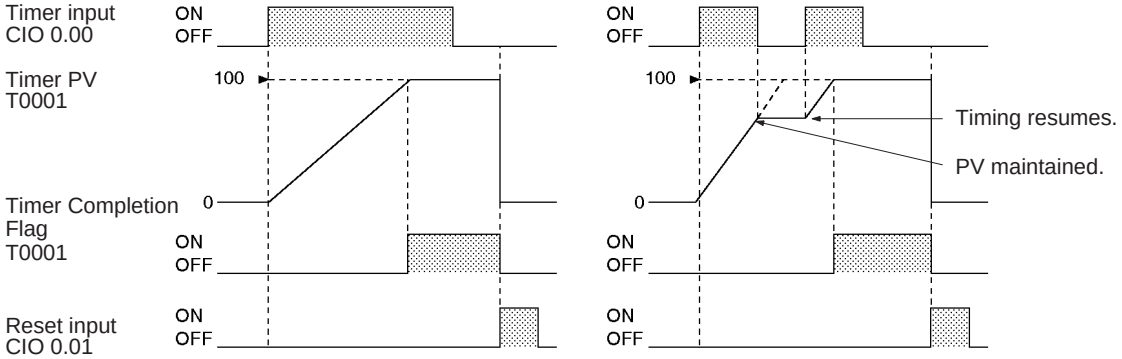
Example

When timer input CIO 0.00 is ON in the following example, the timer PV will begin counting up from 0. Timer Completion Flag T0001 will be turned ON when the PV reaches the SV.

If the reset input is turned ON, the timer PV will be reset to 0000 and the Completion Flag (T0001) will be turned OFF. (Usually the reset input is turned ON to reset the timer and then the timer input is turned ON to start timing.)

If the timer input is turned OFF before the SV is reached, the timer will stop timing but the PV will be maintained. The timer will resume from its previous PV when the timer input is turned ON again.





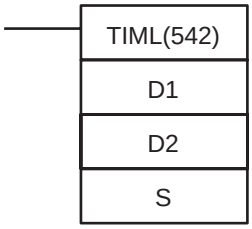
3-5-5 LONG TIMER: TIML(542)/TIMLX(553)

Purpose

TIML(542)/TIMLX(553) operates a decrementing timer with units of 0.1 s that can time up to 115 days for TIML(542) and 4,971 days for TIMLX(543). The timer accuracy is 0 to 0.01 s.

Ladder Symbol

BCD

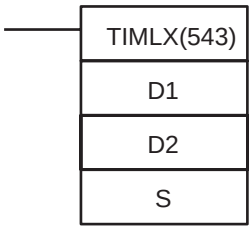


D1: Completion Flag

D2: PV word

S: SV word

Binary



D1: Completion Flag

D2: PV word

S: SV word

Variations

Variations		TIML(542)/ TIMLX(553)
	Executed Each Cycle for ON Condition	
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

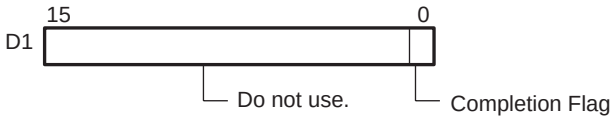
Applicable Program Areas

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

Operands

D1: Completion Flag

Bit 0 of D1 acts as the Completion Flag for TIML(542)/TIMLX(553).



D2: PV Word

D2+1 and D2 contain the 8-digit binary or BCD PV. (D2 and D2+1 must be in the same data area.) The PV can range from #00000000 to #99999999 for TIML(542) and &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hexadecimal) for TIMLX(553).

S: SV Word

S+1 and S contain the 8-digit binary or BCD SV. (S and S+1 must be in the same data area.) The SV must be between #00000000 to #99999999 for TIML(542) and &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hexadecimal) for TIMLX(553).

Operand Specifications

Area	D1	D2	S
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	
Work Area	W0 to W511	W0 to W510	
Holding Bit Area	H0 to H511	H0 to H510	
Auxiliary Bit Area	A448 to A959	A448 to A958	A0 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	D0 to D32767	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		BCD: #00000000 to 99999999 (BCD) "&" cannot be used. Binary: &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hex)
Data Registers	---		

Area	D1	D2	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 IR15		

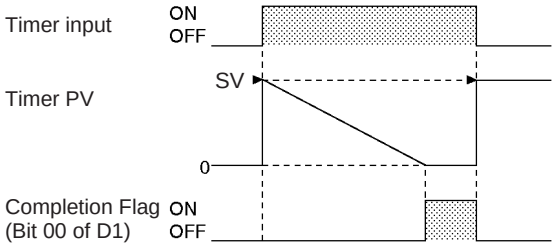
Description

TIML(542)/TIMLX(553) is a decrementing ON-delay timer with units of 0.1-s that uses an 8-digit SV and an 8-digit PV.

When the timer input is OFF, the timer 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, TIML(542)/TIMLX(553) starts decrementing the PV in D2+1 and D2. 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 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 the PV contained in D2+1 and D2 is not BCD. ON if the SV contained in S+1 and S is not BCD. OFF in all other cases.

Precautions

Unlike most timers, TIML(542)/TIMLX(553) does not use a timer number. (Timer area PV refreshing is not performed for TIML(542)/TIMLX(553).)

Since the Completion Flag for TIML(542)/TIMLX(553) is in a data area it can be forced set or forced reset like other bits, but the PV will not change.

The timer's PV is refreshed only when TIML(542)/TIMLX(553) is executed, so the timer will not operate properly when the cycle time exceeds 100 ms because the timer increments in 100-ms units.

The timer's Completion Flag is refreshed only when TIML(542)/TIMLX(553) 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.

When TIML(542)/TIMLX(553) 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.

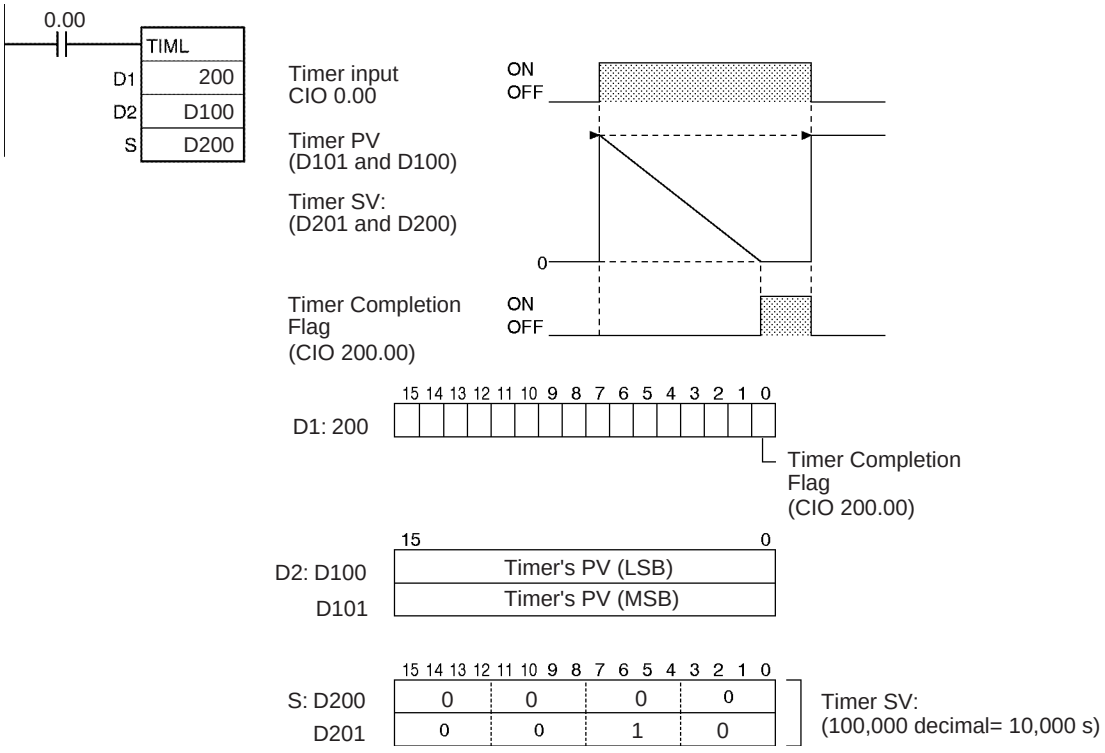
When an operating TIML(542)/TIMLX(553) timer is in a program section between JMP(004) and JME(005) and the program section is jumped, the PV will retain its previous value. Be sure to take this fact into account when TIML(542)/TIMLX(553) is programmed between JMP(004) and JME(005).

Be sure that the words specified for the Completion Flag and PV (D1, D2, and D2+1) are not used in other instructions. If these words are affected by other instructions, the timer might not time out properly.

Example

When timer input CIO 0.00 is ON in the following example, the timer PV (in D101 and D100) will be set to the SV (in D201 and D200) and the PV will begin counting down. The timer Completion Flag (CIO 200.00) will be turned ON when the PV reaches 0000 0000.

When CIO 0.00 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



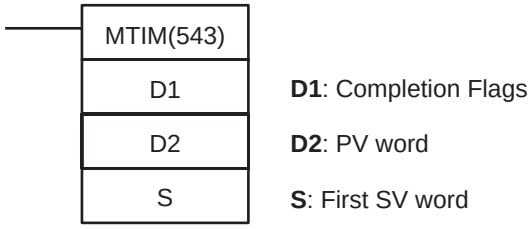
3-5-6 MULTI-OUTPUT TIMER: MTIM(543)/MTIMX(554)

Purpose

MTIM(543)/MTIMX(554) operates a 0.1-s incrementing timer with eight independent SVs and Completion Flags. The set value is 0 to 999.9 s for MTIM(543) and 0 to 6,553.5 s for MTIMX(554), and the timer accuracy is 0 to 0.01 s.

Ladder Symbol

BCD

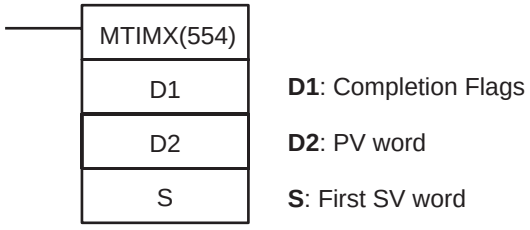


D1: Completion Flags

D2: PV word

S: First SV word

Binary



D1: Completion Flags

D2: PV word

S: First SV word

Variations

Variations		
	Executed Each Cycle for ON Condition	MTIM(543)/ MTIMX(554)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

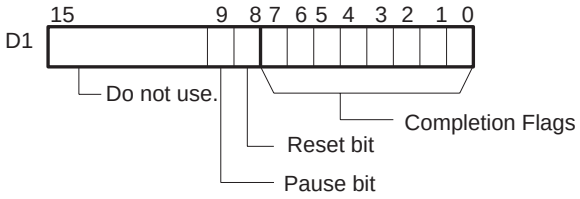
Applicable Program Areas

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

Operands

D1: Completion Flags

D1 contains the eight Completion Flags as well as the pause and reset bits.



D2: PV Word

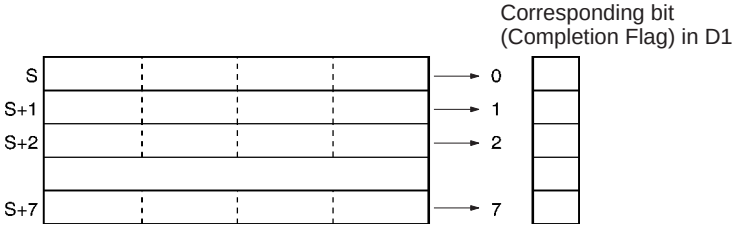
D2 contains the 4-digit binary or BCD PV.

Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)

S: First SV Word

S through S+7 contain the eight independent SVs. Each SV must be as follows:

Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)



Data	Range
BCD	One word for each of 8 timer SV: #0000 to #9999
Binary	One word for each of 8 timer SV: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Note S through S+7 must be in the same data area.

Operand Specifications

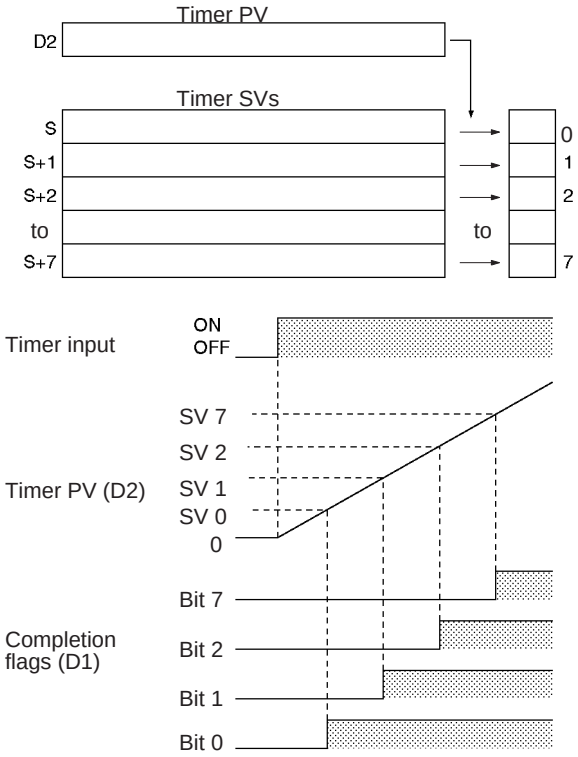
Area	D1	D2	S
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6136
Work Area	W0 to W511		W0 to W504
Holding Bit Area	H0 to H511		H0 to H504
Auxiliary Bit Area	A448 to A959		A0 to A952
Timer Area	T0000 to T4095		T0000 to T4088
Counter Area	C0000 to C4095		C0000 to C4088
DM Area	D0 to D32767		D0 to D32760
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the execution condition for MTIM(543)/MTIMX(554) is ON and the reset and timer bits are both OFF, MTIM(543)/MTIMX(554) increments the PV in D2. If the pause bit is turned ON, the timer will stop incrementing the PV, but the PV will retain its value. MTIM(543)/MTIMX(554) will resume timing when the pause bit goes OFF again.

The PV (content of D2) is compared to the eight SVs in S through S+7 each time that MTIM(543)/MTIMX(554) is executed, and if any of the SVs is less than or equal to the PV, the corresponding Completion Flag (D1 bits 00 through 07) is turned ON.

When the PV reaches 9999, the PV will be reset to 0000 and all of the Completion Flags will be turned OFF. If the reset bit is turned ON while the timer is operating or paused, the PV will be reset to 0000 and all of the Completion Flags will be turned OFF.



The following table shows the operation of MTIM(543)/MTIMX(554) for the four possible combinations of the reset and pause bits.

Reset bit (Bit 08)	Pause bit (Bit 09)	Operation
OFF	OFF	The PV will be updated and the corresponding Completion Flag will be turned ON when SV ≤ PV.
	ON	The PV will not be updated and MTIM(543)/MTIMX(554) will be treated as NOP(000).
ON	OFF	The PV will be reset to 0000 and the Completion Flags will be turned OFF. The PV will not be updated.
	ON	

The reset and pause bits are effective only when the execution condition for MTIM(543)/MTIMX(554) is ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the PV contained in D2 is not BCD. OFF in all other cases.

Precautions

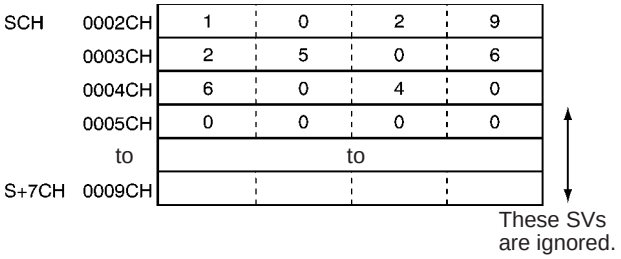
Unlike most timers, MTIM(543)/MTIMX(554) does not use a timer number. (Timer area PV refreshing is not performed for MTIM(543)/MTIMX(554).)

When the PV reaches 9999, the PV will be reset to 0000 and all of the Completion Flags will be turned OFF.

If in BCD mode and an SV in S through S+7 does not contain BCD data, that SV will be ignored. An error will not occur and the Error Flag will not be turned ON.

Since the Completion Flag for MTIM(543)/MTIMX(554) is in a data area it can be forced set or forced reset like other bits, but the PV will not change.

When eight or fewer SVs are required, set the word after the last SV to 0000. MTIM(543)/MTIMX(554) will ignore the SV that is set to 0000 and all of the remaining SVs.



The timer's PV is refreshed only when MTIM(543)/MTIMX(554) is executed, so the timer will not operate properly when the cycle time exceeds 100 ms because the timer increments in 100-ms units. To ensure precise timing and prevent problems caused by long cycle times, input the same MTIM(543)/MTIMX(554) instruction at several points in the program.

The timer's Completion Flag is refreshed only when MTIM(543)/MTIMX(554) 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.

When MTIM(543)/MTIMX(554) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will retain its previous value (it will not be reset). Be sure to take this fact into account when MTIM(543)/MTIMX(554) is programmed between IL(002) and ILC(003).

When an operating MTIM(543)/MTIMX(554) timer is in a program section between JMP(004) and JME(005) and the program section is jumped, the PV will retain its previous value. Be sure to take this fact into account when MTIM(543)/MTIMX(554) is programmed between JMP(004) and JME(005).

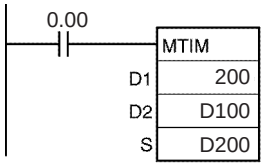
Be sure that the words specified for the Completion Flags and PV (D1 and D2) are not used in other instructions. If these words are affected by other instructions, the timer might not time out properly.

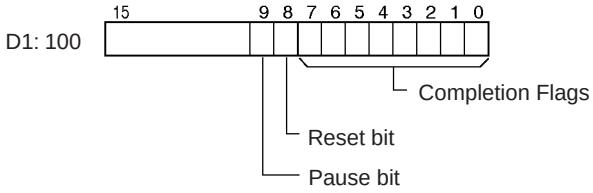
If a word in the CIO area is specified for D1, the SET and RSET instructions can be used to control the pause and reset bits.

Example

When CIO 0.00 is ON and the pause bit (CIO 200.09) is OFF in the following example, the timer will start operating when the reset bit (CIO 200.08) is turned from ON to OFF. The timer's PV will begin timing up from 0000.

The eight SVs in D200 through D207 are compared to the PV and the corresponding Completion Flags (CIO 200.00 through CIO 200.07) are turned ON when the SV ≤ PV.





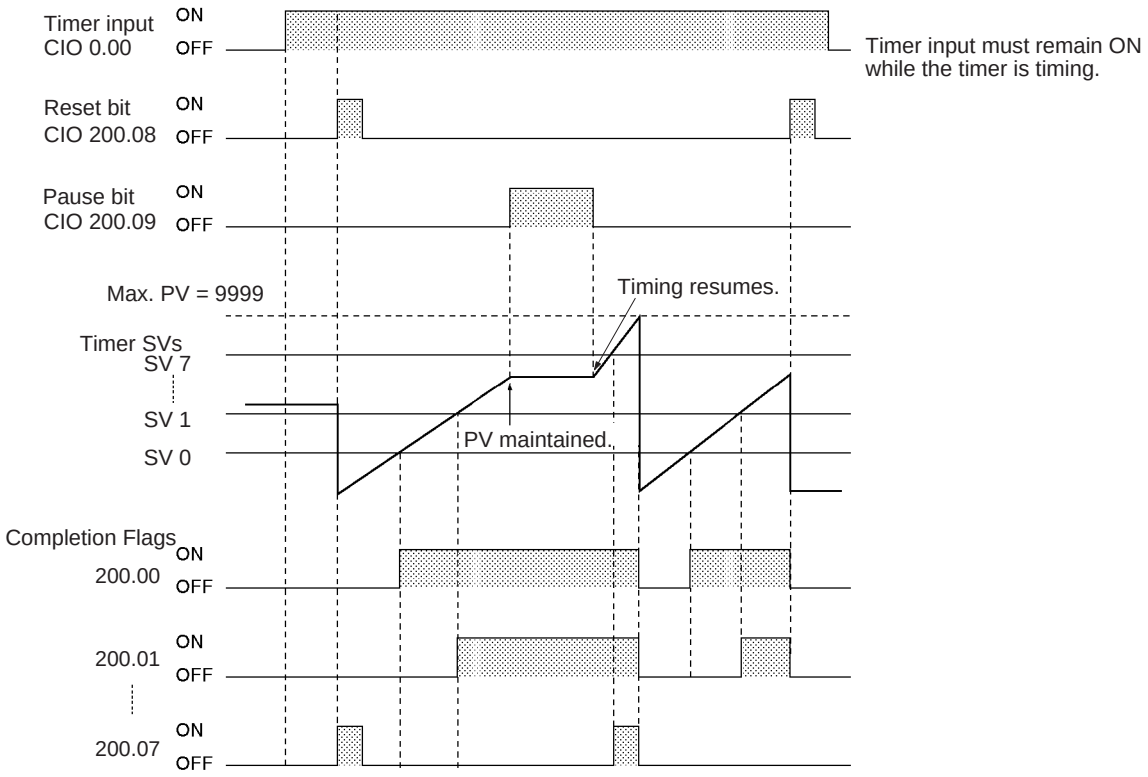
Timer PV (D2: D100) — 15 … 0 (Incrementing)

0	1	0	0

Corresponding completion flag ON when SV ≤ PV.

Timer SVs (15 … 0)

S: D200	0	0	8	0	1 0
S+1: D201	0	0	9	0	1 1
S+2: D202	0	1	0	0	1 2
S+3: D203	0	1	1	0	0 3
S+4: D204	0	1	2	0	0 4
S+5: D205	0	1	3	0	0 5
S+6: D206	0	1	5	0	0 6
S+7: D207	1	0	0	0	0 7



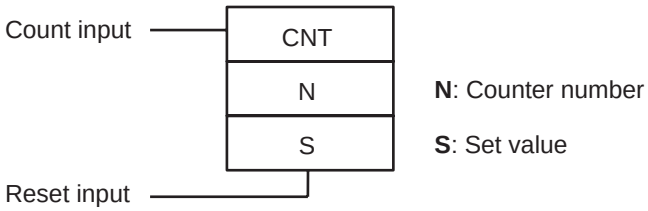
3-5-7 COUNTER: CNT/CNTX(546)

Purpose

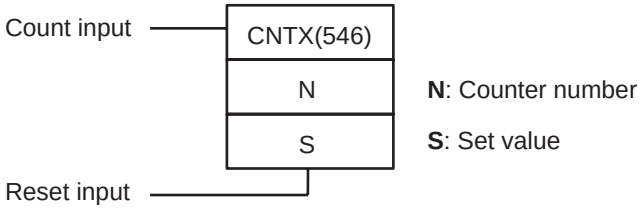
CNT/CNTX(546) operates a decrementing counter. The setting range 0 to 9,999 for CNT and 0 to 65,535 for CNTX(546).

Ladder Symbol

BCD



Binary



Variations

Variations		
	Executed Each Cycle for ON Condition	CNT/ CNTX(546)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0000 and 4095 (decimal).

S: Set Value

Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	---	T0000 to T4095
Counter Area	0000 to 4095 (decimal)	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767

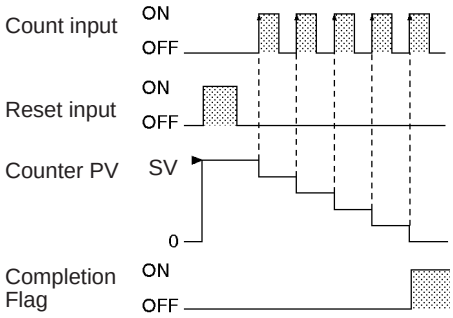
Area	N	S
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

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 or by using the CNR(545)/CNRX(547) instruction. 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 in BCD mode and S does not contain BCD data. OFF in all other cases.
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

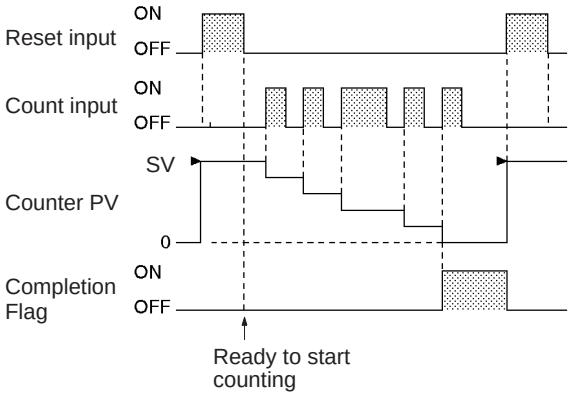
Precautions

Counter numbers are shared by the CNT, CNTX(546), CNTR(012), CNTRX(548), CNTW(814), and CNTWX(818) instructions. If two counters share the same counter number but are not used simultaneously, a duplication error will be generated when the program is checked but the counters will operate normally. Counters which share the same counter number will not operate properly if they are used simultaneously.

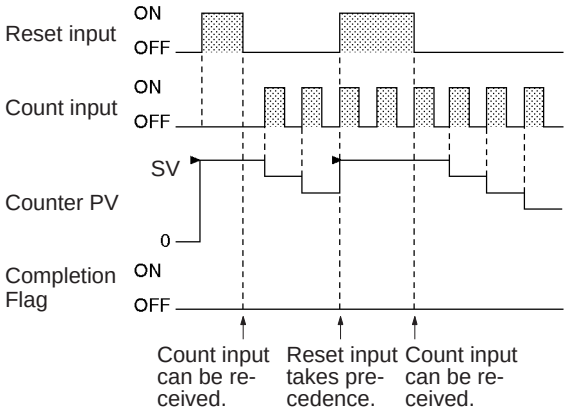
A counter’s PV is refreshed when the count input goes from OFF to ON and the Completion Flag is refreshed each time that CNT/CNTX(546) is executed. The Completion Flag is turned ON if the PV is 0 and it is turned OFF if the PV is not 0.

When a CNT/CNTX(546) counter is forced set, its Completion Flag will be turned ON and its PV will be reset to 0000. When a CNT/CNTX(546) 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.



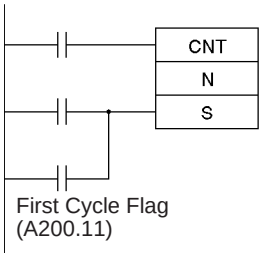
The reset input will take precedence and the counter will be reset if the reset input and count input are both ON at the same time. (The PV will be reset to the SV and the Completion Flag will be turned OFF.)



The operation of the = Flag and N Flag depends on the model of the CPU Unit. Refer to *Flags*, above, for details.

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 retained even through a power interruption. If you want to restart counting from the SV instead of resuming the count from the retained PV, add the First Cycle Flag (A200.11) as a reset input to the counter.



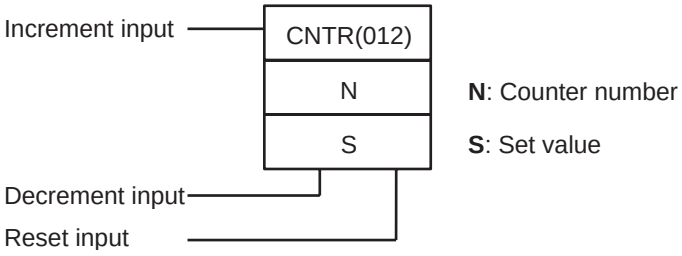
3-5-8 REVERSIBLE COUNTER: CNTR(012)/CNTRX(548)

Purpose

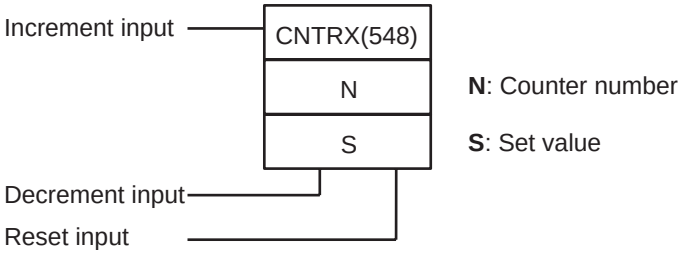
CNTR(012)/CNTRX(548) operates a reversible counter.

Ladder Symbol

BCD



Binary



Variations

Variations		
	Executed Each Cycle for ON Condition	CNTR(012)/ CNTRX(548)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0000 and 4095 (decimal).

S: Set Value

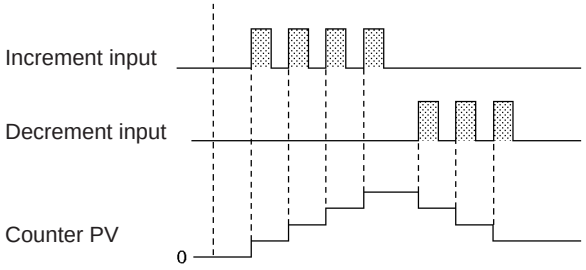
Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)

Operand Specifications

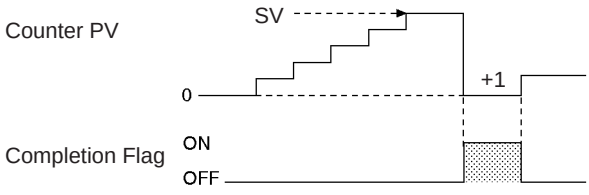
Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	---	T0000 to T4095
Counter Area	0000 to 4095 (decimal)	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
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	

Description

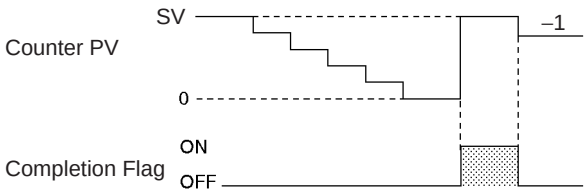
The counter PV is incremented by 1 every time that the increment input goes from OFF to ON and it is decremented by 1 every time that the decrement input goes from OFF to ON. The PV can fluctuate between 0 and the SV.



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 if in BCD mode and S does not contain BCD data. OFF in all other cases.

Precautions

Counter numbers are shared by the CNT, CNTX(546), CNTR(012), CNTRX(548), CNTW(814), and CNTWX(818) instructions. If two counters share the same counter number but are not used simultaneously, a duplication error will be generated when the program is checked but the counters will operate normally. Counters which share the same counter number will not operate properly if they are used 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.

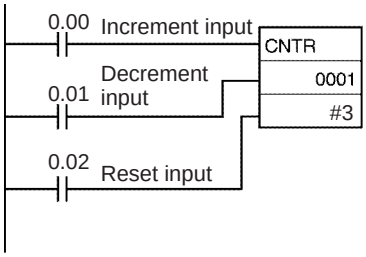
When inputting the CNTR(012)/CNTRX(548) instruction with mnemonics, first enter the increment input (II), then the decrement input (DI), the reset input (R), and finally the CNTR(012)/CNTRX(548) instruction. When entering with the ladder diagrams, first input the increment input (II), then the CNTR(012)/CNTRX(548) instruction, the decrement input (DI), and finally the reset input (R).

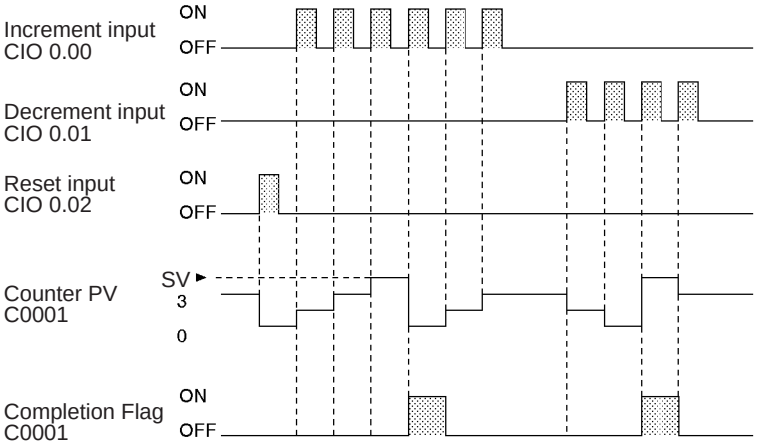
Examples

Basic Operation of CNTR(012)/CNTRX(548)

The counter PV is reset to 0 by turning the reset input (CIO 0.02) ON and OFF. The PV is incremented by 1 each time that the increment input (CIO 0.00) 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.

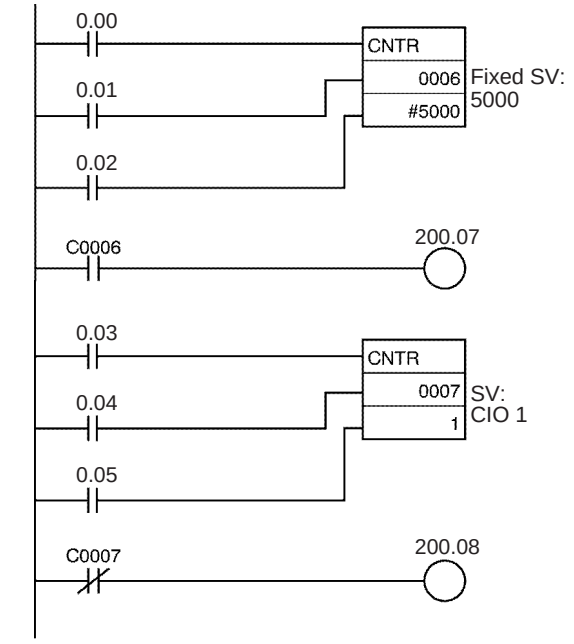
Likewise, the PV is decremented by 1 each time that the decrement input (CIO 0.01) goes from OFF to ON. When the PV is decremented from 0, it is automatically set to the SV (3) and the Completion Flag is turned ON.

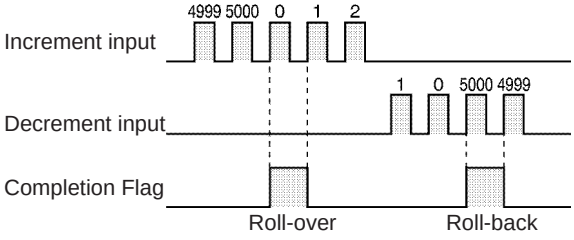




Specifying the SV in a Word

In the following example, the SV for CNTR(012) 0007 is determined by the content of CIO 1. The content of CIO 1 can be controlled by an external switch so that the set value can be changed manually from the switch.





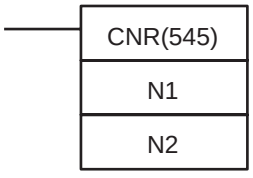
3-5-9 RESET TIMER/COUNTER: CNR(545)/CNRX(547)

Purpose

Resets the timers or counters within the specified range of timer or counter numbers.

Ladder Symbol

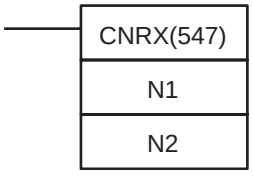
BCD



N_1: First number in range

N_2: Last number in range

Binary



N_1: First number in range

N_2: Last number in range

Variations

Variations	Executed Each Cycle for ON Condition	CNR(545)/ CNRX(547)
	Executed Once for Upward Differentiation	@CNR(545)/ CNRX(547)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N_1: First Number in Range

N_1 must be a timer number between T0000 and T4095 or a counter number between C0000 and C4095.

N_2: Last Number in Range

N_2 must be a timer number between T0000 and T4095 or a counter number between C0000 and C4095.

Note N_1 and N_2 must be in the same data area, i.e., N_1 and N_2 must be timer numbers or counter numbers.

Operand Specifications

Area	N_1	N_2
CIO Area	---	---
Work Area	---	---
Holding Bit Area	---	---
Auxiliary Bit Area	---	---
Timer Area	C0000 to C4095	C0000 to C4095
Counter Area	T0000 to T4095	T0000 to T4095
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 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

CNR(545)/CNRX(547) resets the Completion Flags of all timers or counters from N_1 to N_2. At the same time, the PVs will all be set to the maximum value (9999 for BCD and FFFF for binary). (The PV will be set to the SV the next time that the timer or counter instruction is executed.)

Timers Reset by CNR(545)/CNRX(547)

The following timers will be reset if their timer numbers fall within the specified range: TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TTIM(087), TTIMX(555), TIMW(813), TIMWX(816), TMHW(815), and TMHWX(817). When a timer is reset, its Completion Flag is turned OFF and its PV is set to the maximum value of 9999.

Note The TIML(542), TIMLX(553), MTIM(543), and MTIMX(554) timers are not reset by CNR(545)/CNRX(547) because these timers do not use timer numbers.

Counters Reset by CNR(545)/CNRX(547)

The following counters will be reset if their counter numbers fall within the specified range: CNT, CNTX(546), CNTR(012), CNTRX(548), CNTW(814), and CNTWX(818). When a counter is reset, its Completion Flag is turned OFF and its PV is set to the maximum value of 9999.

Flags

Name	Label	Operation
Error Flag	ER	ON if N_1 is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer or counter. ON if N_2 is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer or counter. ON if N_1 and N_2 are not in the same data area. OFF in all other cases.

Precautions

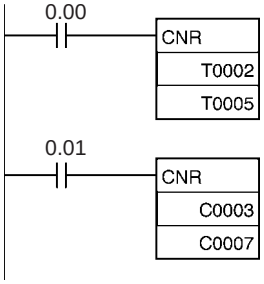
CNR(545)/CNRX(547) does not reset the timer/counter instructions themselves, it resets the PVs and Completion Flags allocated to those instructions. In most cases, the effect of CNR(545)/CNRX(547) is different from directly resetting the instructions. For example, when a TIM/TIMX(550) instruction is reset directly its PV is set to the SV, but when that timer is reset by CNR(545)/CNRX(547) its PV is set to the maximum value of 9999.

When N1 and N2 are specified with N1>N2, only the Completion Flag for the timer/counter number will be reset.

Example

When CIO 0.00 is ON in the following example, the Completion Flags for timers T0002 to T0005 are turned OFF and the timers' PVs are set to the maximum value of 9999.

When CIO 0.01 is ON, the Completion Flags for counters C0003 to C0007 are turned OFF and the counters' PVs are set to the maximum value of 9999.



3-5-10 Example Timer and Counter Applications

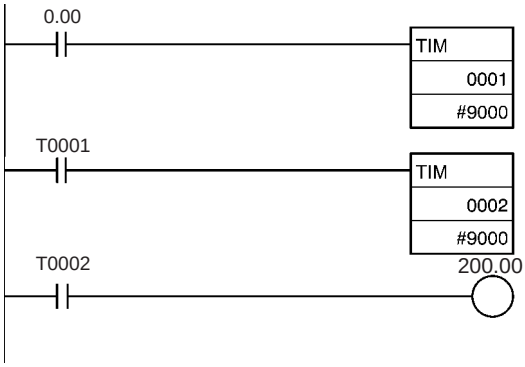
The following examples show various applications of timer and counter instructions including long-term timers, a two-stage counter, ON/OFF delay, one-shot bit, and flicker bit.

Example 1: Long-term Timers

The following program examples show three ways to create long-term timers with standard TIM and CNT instructions.

Two TIM Instructions

In this example, two TIM instructions are combined to make a 30-minute timer.

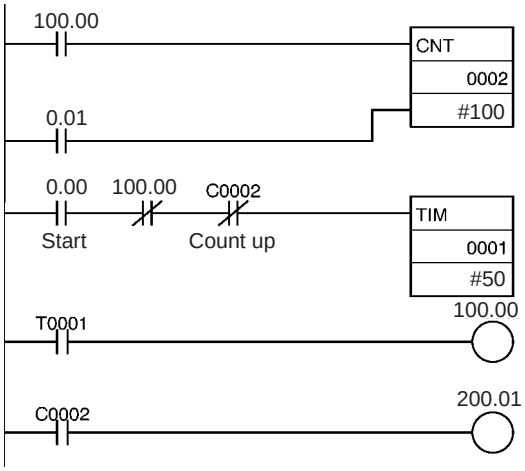


Address	Instruction	Operands
000000	LD	0.00
000001	TIM	0001
		#9000
000002	LD	T0001
000003	TIM	0002
		#9000
000004	LD	T0002
000005	OUT	200.00

TIM and CNT Instructions

In this example, a TIM instruction and a CNT instruction are combined to make a 500-second timer.

TIM 0001 generates a pulse every 5 s and CNT 0002 counts these pulses. The set value for this combination is the timer interval × counter SV. In this case, the timer SV would be 5 s × 100 = 500 s. With this combination, the long-term timer's PV is actually the PV of a counter, which is maintained through power interruptions.

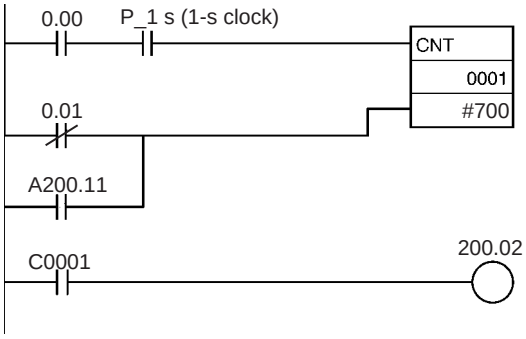


Address	Instruction	Operands
000000	LD	100.00
000001	LD	0.01
000002	CNT	0002
		#100
000003	LD	0.00
000004	AND NOT	100.00
000005	AND NOT	C0002
000006	TIM	0001
		#50
000007	LD	T0001
000008	OUT	100.00
000009	LD	C0002
000010	OUT	201.01

Clock Pulse and CNT Instruction

In this example, a CNT instruction counts the pulses from the 1-s clock pulse to make a 700-second timer.

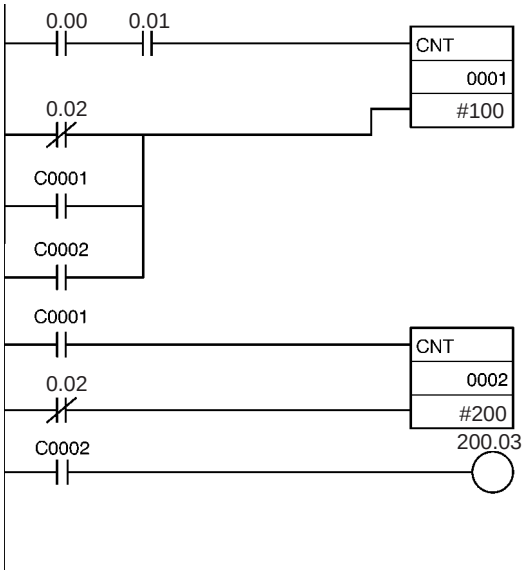
If the First Cycle Flag (A200.11) is ORed with the counter's reset input (CIO 0.01), the counter's PV will be reset to the SV (0700) when program execution begins rather than resuming the count from the previous PV.



Address	Instruction	Operands
000000	LD	0.00
000001	AND	1 s
000002	LD NOT	0.01
000003	OR	A200.11
000004	CNT	0001
		#700
000005	LD	C0001
000006	OUT	200.02

Example 2: Two-stage Counter

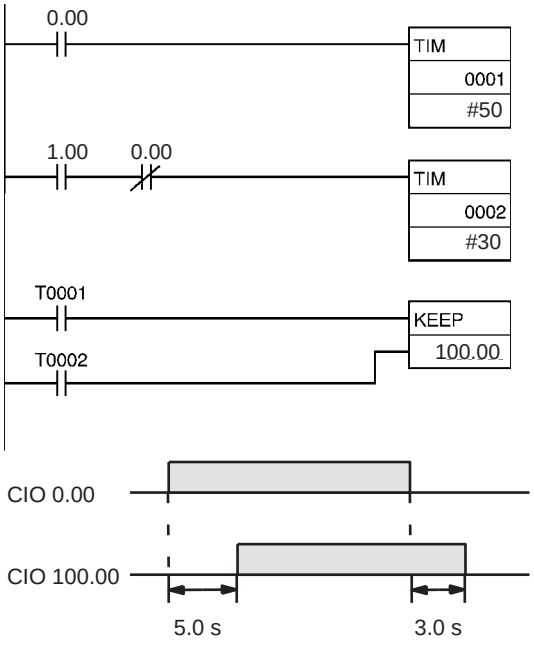
When an SV higher than 9999 is required, two counters can be combined as shown in the following example. In this case, two CNT instructions are combined to make a BCD counter with an SV of 20,000.



Address	Instruction	Operands
000000	LD	0.00
000001	AND	0.01
000002	LD NOT	0.02
000003	OR	C0001
000004	OR	C0002
000005	CNT	0001
		#100
000006	LD	C0001
000007	LD NOT	0.02
000008	CNT	0002
		#200
000009	LD	C0002
000010	OUT	200.03

Example 3: ON/OFF Delay

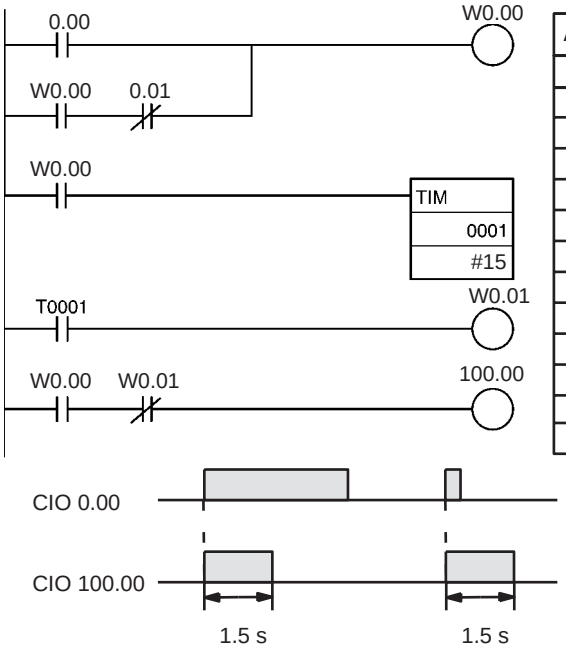
In this example two TIM timers are combined with KEEP(011) to make an ON delay and an OFF delay. CIO 100.00 will be turned ON 5.0 seconds after CIO 0.00 goes ON and it will be turned OFF 3.0 seconds after CIO 0.00 goes OFF.



Address	Instruction	Operands
000000	LD	0.00
000001	TIM	0001
		#50
000002	LD	1.00
000003	AND NOT	0.00
000004	TIM	0002
		#30
000005	LD	T0001
000006	LD	T0002
000007	KEEP(011)	100.00

Example 4: One-shot Bit

A TIM timer can be combined with OUT or OUT NOT to control how long a particular bit is ON or OFF. In this example, CIO 100.00 will be ON for 1.5 seconds (the SV of T0001) after CIO 0.00 goes ON.



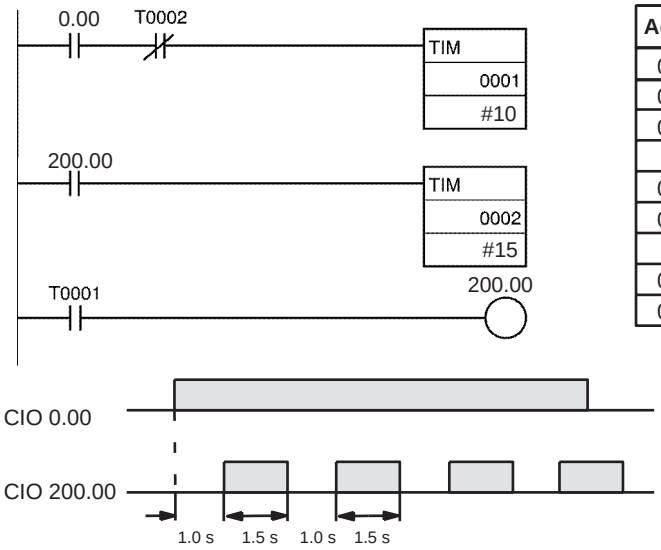
Address	Instruction	Operands
000000	LD	0.00
000001	LD	W0.00
000002	AND NOT	0.01
000003	OR LD	---
000004	OUT	W0.00
000005	LD	W0.00
000006	TIM	0001
		#15
000007	LD	T0001
000008	OUT	W0.01
000009	LD	W0.00
000010	AND NOT	W0.01
000011	OUT	100.00

Example 5: Flicker Bit

The following program examples show two ways to create flicker bits. The second example just mimics a clock pulse.

Two TIM Instructions

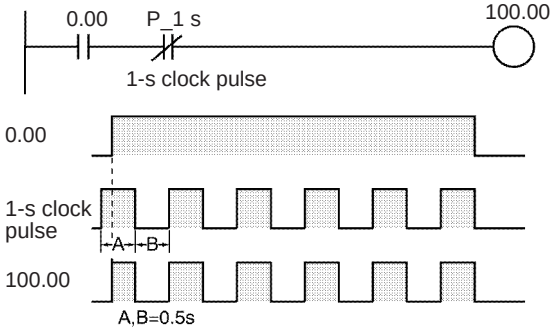
Two TIM timers can be combined to make a bit turn ON and OFF at regular intervals while the execution condition is ON. In this example, CIO 200.00 will be OFF for 1.0 second and then ON for 1.5 seconds as long as CIO 0.00 is ON.



Address	Instruction	Operands
000000	LD	0.00
000001	AND	T0002
000002	TIM	0001
		#10
000003	LD	200.00
000004	TIM	0002
		#15
000005	LD	T0001
000006	OUT	200.00

Clock Pulse

The desired execution condition can be combined with a clock pulse to mimic the clock pulse (0.1 s, 0.2 s, or 1.0 s).



Address	Instruction	Operands
000000	LD	0.00
000001	AND	1s
000002	OUT	100.00

3-5-11 Indirect Addressing of Timer/Counter Numbers

Timer and counter numbers can be indirectly addressed using Index Registers. When Index Registers will be used for indirect addressing, use MOVRW(561) (MOVE TIMER/COUNTER PV TO REGISTER) to set the PLC memory address of the desired timer or counter's PV to the desired Index Register.

The following timers and counters can be indirectly addressed using Index Registers: TIM, TIMX(550), TIMH(015), TIMHX(551), TTIM(087), TTIMX(555), TMHH(540), TMHHX(552), TIMW(813), TIMWX(816), TMHW(815), TMHWX(817), CNT, CNTX(546), CNTR(012), CNTRX(548), CNTW(814), and CNTWX(818). (These are the timers and counters that use timer and counter numbers.)

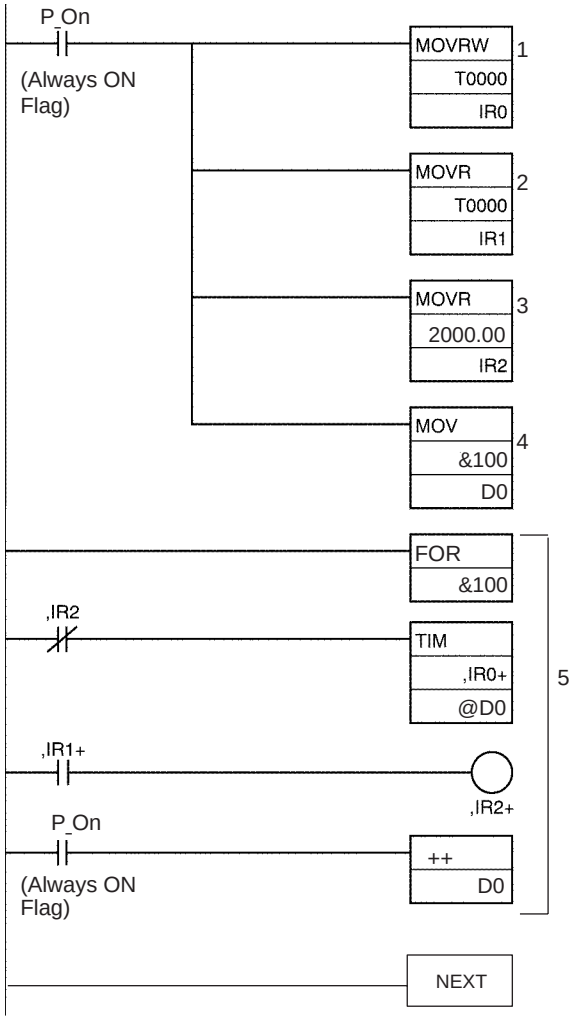
The timer or counter instruction will not be executed if the PLC memory address in the specified Index Register is not the address of a timer or counter PV.

Using Index Registers to indirectly address timers and counters can reduce the size of the program and increase flexibility. For example, common subroutines can be created.

Example

The following example shows a program section that uses indirect addressing to define and start 100 timers with SVs contained in D100 through D199. IR0 contains the PLC memory address of the timer PV and IR1 contains the PLC memory address of the timer Completion Flag.

DM address	Content	Function
D100	0010	SV for T0000
D101	0100	SV for T0001
D102	0050	SV for T0002
. . .	. . .	. . .
D199	0999	SV for T0099



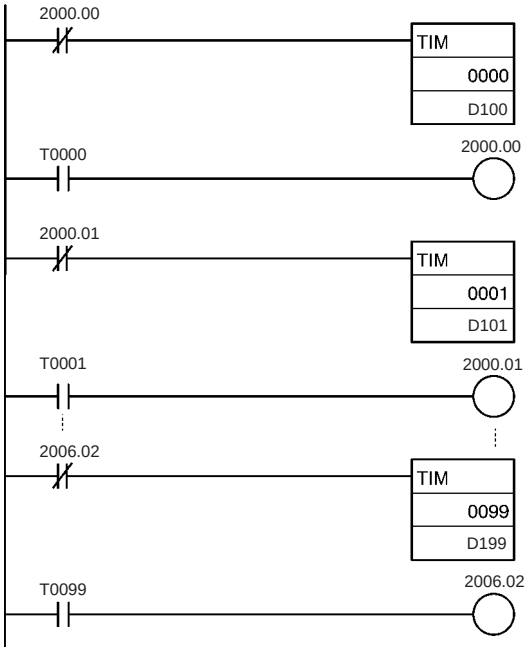
1,2,3... 1. MOVRW(561) moves the PLC memory address of the PV for timer T0000 to IR0. Afterwards IR0 can be used in place of the timer number.

2. MOVR(560) moves the PLC memory address of the Completion Flag for timer T0000 to IR1.
3. MOVR(560) moves the PLC memory address of CIO 2000.00 into IR2.
4. MOV(021) moves &100 into D0 for indirect addressing of the timer SVs.
5. The content of IR0, IR1, IR2, and D0 are incremented by 1 each time as this loop is executed 100 times, starting timers T0 through T99.

The loop in the program above has 4 input parameters which are used to start all 100 timers with this common subroutine.

IR0 The PLC memory address of the timer's PV
IR1 The PLC memory address of the timer's Completion Flag
IR2 The PLC memory address of the timer's execution condition
D0 The DM address of the word containing the timer's SV

The subroutine above is equivalent to the 400 instructions below.



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	210
Time Comparison Instructions	=DT, <>DT, <DT, <=DT, >DT, >=DT (LD, AND, OR)	341 to 346	216
COMPARE	CMP	020	221
DOUBLE COMPARE	CMPL	060	223
SIGNED BINARY COMPARE	CPS	114	226
DOUBLE SIGNED BINARY COMPARE	CPSL	115	228
MULTIPLE COMPARE	MCMP	019	231
TABLE COMPARE	TCMP	085	234
BLOCK COMPARE	BCMP	068	236
EXPANDED BLOCK COMPARE	BCMP2	502	239

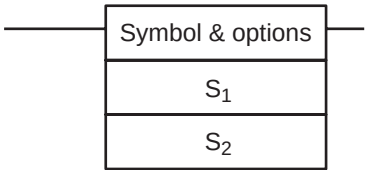
3-6-1 Input Comparison Instructions (300 to 328)

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 and *3-15-21 Double-precision Floating-point Input Instructions* for details on double-precision floating-point input comparison instructions.

Ladder Symbol



Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction
Immediate Refreshing Specification		Not supported

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 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	
Timer Area	T0000 to T4095	

Area	S_1	S_2
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal)	
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	

Operand Specifications for Instructions for Double-length Data

Area	S_1	S_2
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal)	
Data Registers	---	
Index Registers	IR0 to IR15 (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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

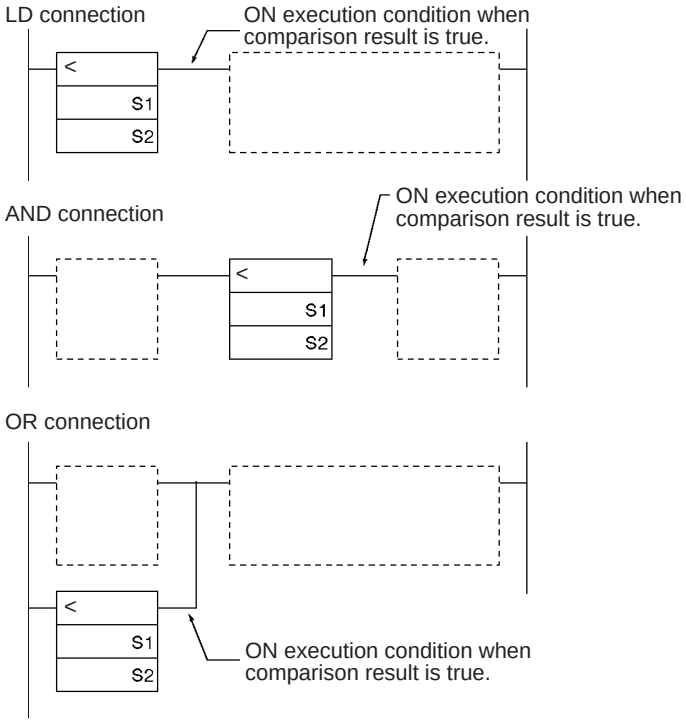
Description

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 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	

Code	Mnemonic	Name	Function
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	
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.

Name	Label	Operation
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.

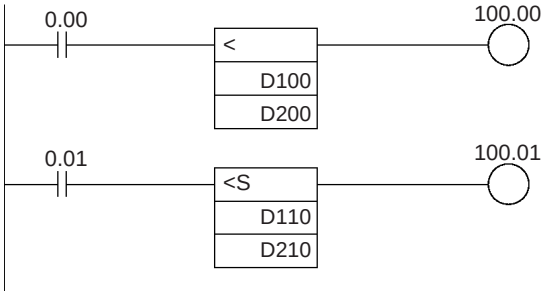
Precautions

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

Examples

AND LESS THAN: AND<(310)

When CIO 0.00 is ON in the following example, the contents of D100 and D200 are compared in as unsigned binary data. If the content of D100 is less than that of D200, CIO 100.00 is turned ON and execution proceeds to the next line. If the content of D100 is not less than that of D200, the remainder of the instruction line is skipped and execution moves to the next instruction line.



Unsigned LESS THAN Comparison

S_1: D100 — 8714 — Decimal: 34,580

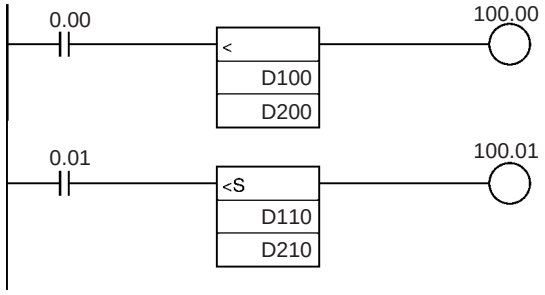
S_2: D200 — 3A1C — Decimal: 14,876

34,580 > 14,876

(Will not proceed to next line.)

AND SIGNED LESS THAN: AND<S(312)

When CIO 0.01 is ON in the following example, the contents of D110 and D210 are compared as signed binary data. If the content of D110 is less than that of D210, CIO 100.01 is turned ON and execution proceeds to the next line. If the content of D110 is not less than that of D210, the remainder of the instruction line is skipped and execution moves to the next instruction line.



Signed LESS THAN Comparison

S_1: D110 — 8714 — Decimal: –30,956

S_2: D210 — 3A1C — Decimal: 14,876

–30,956 < 14,876

(Will proceed to next line.)

3-6-2 Time Comparison Instructions (341 to 346)

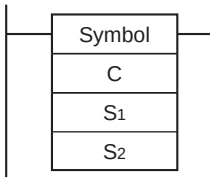
Purpose

Time comparison instructions compare two BCD time values and create an ON execution condition when the comparison condition is true.

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

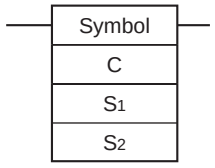
Ladder Symbol

LD



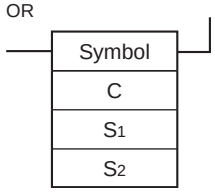
C: Control word
S1: First word of present time
S2: First word of comparison time

AND



C: Control word
S1: First word of present time
S2: First word of comparison time

OR



C: Control word
S1: First word of present time
S2: First word of comparison time

Variations

Variations	Creates ON Each Cycle Comparison is True	Time comparison instruction
Immediate Refreshing Specification		Not supported

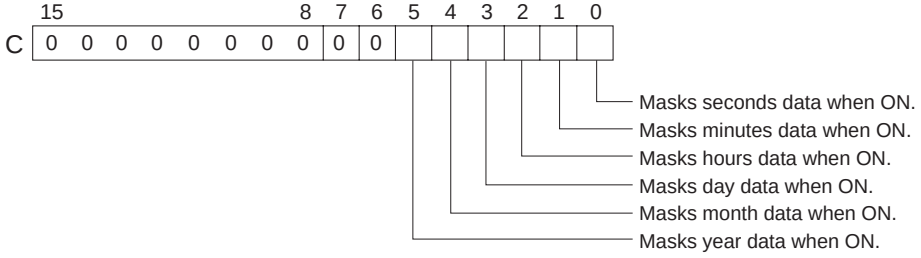
Applicable Program Areas

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

Operands

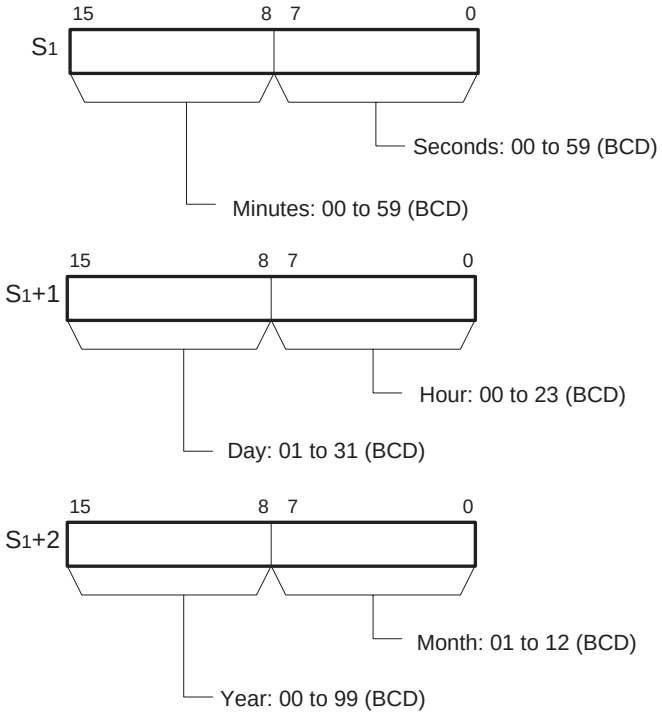
C: Control Word

Bits 00 to 05 of C specify whether or not the time data will be masked for the comparison. Bits 00 to 05 mask the seconds, minutes, hours, day, month, and year, respectively. If all 6 values are masked, the instruction will not be executed, the execution condition will be OFF, and the Error Flag will be turned ON.



S_1 through S_1+2: Present Time Data

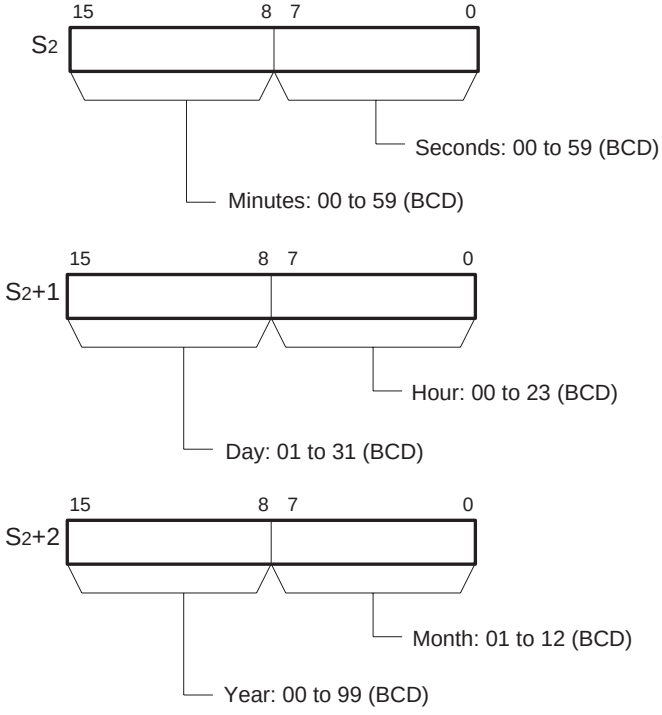
S_1 through S_1+2 contain the present time data. S_1 through S_1+2 must be in the same data area.



Note When using the CPU Unit's internal clock data for the comparison, set S_1 to A351 to specify the CPU Unit's internal clock data (A351 to A353).

S_2 through S_2+2: Comparison Time Data

S_2 through S_2+2 contain the comparison time data. S_2 through S_2+2 must be in the same data area.



Operand Specifications

Area	C	S_1	S_2
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6141	CIO 0 to CIO 6142
Work Area	W0 to W511	W0 to W509	W0 to W510
Holding Bit Area	H0 to H511	H0 to H509	H0 to H510
Auxiliary Bit Area	A448 to A959	A0 to A957	A0 to A958
Timer Area	T0000 to T4095	T0000 to T4093	T0000 to T4094
Counter Area	C0000 to C4095	C0000 to C4093	C0000 to C4094
DM Area	D0 to D32767	D0 to D32765	D0 to D32766
Indirect DM addresses in binary	---	@ D0 to @ D32767	
Indirect DM addresses in BCD	---	*D0 to *D32767	
Constants	See previous page.	See previous page.	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

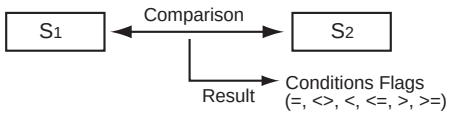
The time comparison instruction compares the unmasked values (corresponding bit of C set to 0) of the present time data in S_1 to S_1+2 with the comparison time data in S_2 to S_2+2 and creates an ON execution condition when the comparison condition is true. At the same time, the result of a time comparison instruction is reflected in the arithmetic flags (=, <>, <, <=, >, >=).

There are 18 possible combinations of time comparison instructions.

Any time values that are masked in the control word (C) are not included in the comparison.

The following table shows the ON/OFF status of each flag for each comparison result.

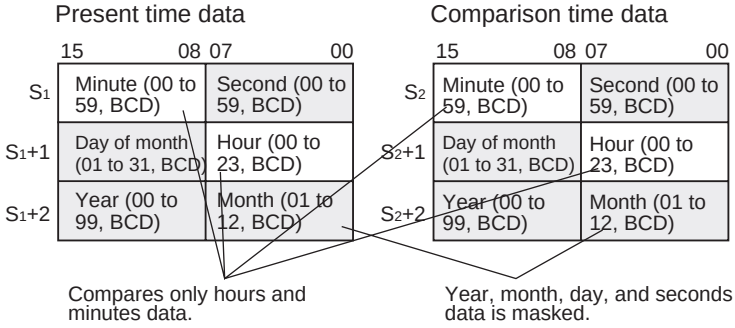
Result	Flag status					
	=	<>	<	<=	>	>=
$S_1 = S_2$	ON	OFF	OFF	ON	OFF	ON
$S_1 > S_2$	OFF	ON	OFF	OFF	ON	ON
$S_1 < S_2$	OFF	ON	ON	ON	OFF	OFF



Masking Time Values

Time values can be masked individually and excluded from the comparison operation. To mask a time value, set the corresponding bit in the control word (C) to 1. Bits 00 to 05 of C mask the seconds, minutes, hours, day, month, and year, respectively.

Example:
When C = 39 hex, the rightmost 6 bits are 111001 (year=1, month=1, day=1, hours=0, minutes=0, and seconds=1) so only the hours and minutes are compared. This mask setting can be used to perform a particular operation at a given time (hour and minute) each day.



Previous data comparison instructions compared data in 16-bit units. The time comparison instructions are limited to comparing 8-bit time values.

The following table shows the structure of the CPU Unit's internal Calendar/Clock Area.

Addresses	Contents
A351.00 to A351.07	Second (00 to 59, BCD)
A351.08 to A351.15	Minute (00 to 59, BCD)
A352.00 to A352.07	Hour (00 to 23, BCD)
A352.08 to A352.15	Day of month (01 to 31, BCD)
A353.00 to A353.07	Month (01 to 12, BCD)
A353.08 to A353.15	Year (00 to 99, BCD)

The Calendar/Clock Area can be set with the CX-Programmer, DATE(735) instruction, or "CLOCK WRITE" FINS command (0702 hex).

Summary of Time Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 18 time comparison instructions.

Code	Mnemonic	Name	Function
341	LD= DT	LOAD EQUAL	True if S1 = S2
	AND=DT	AND EQUAL	
	OR=DT	OR EQUAL	
342	LD<>DT	LOAD NOT EQUAL	True if S1 ≠ S2
	AND<>DT	AND NOT EQUAL	
	OR<>DT	OR NOT EQUAL	
343	LD<DT	LOAD LESS THAN	True if S1 < S2
	AND<DT	AND LESS THAN	
	OR<DT	OR LESS THAN	
344	LD<=DT	LOAD LESS THAN OR EQUAL	True if S1 ≤ S2
	AND<=DT	AND LESS THAN OR EQUAL	
	OR<=DT	OR LESS THAN OR EQUAL	
345	LD>DT	LOAD GREATER THAN	True if S1 > S2
	AND>DT	AND GREATER THAN	
	OR>DT	OR GREATER THAN	

Code	Mnemonic	Name	Function
346	LD>=DT	LOAD GREATER THAN OR EQUAL	True if S1 ≥ S2
	AND>=DT	AND GREATER THAN OR EQUAL	
	OR>=DT	OR GREATER THAN OR EQUAL	

Flags

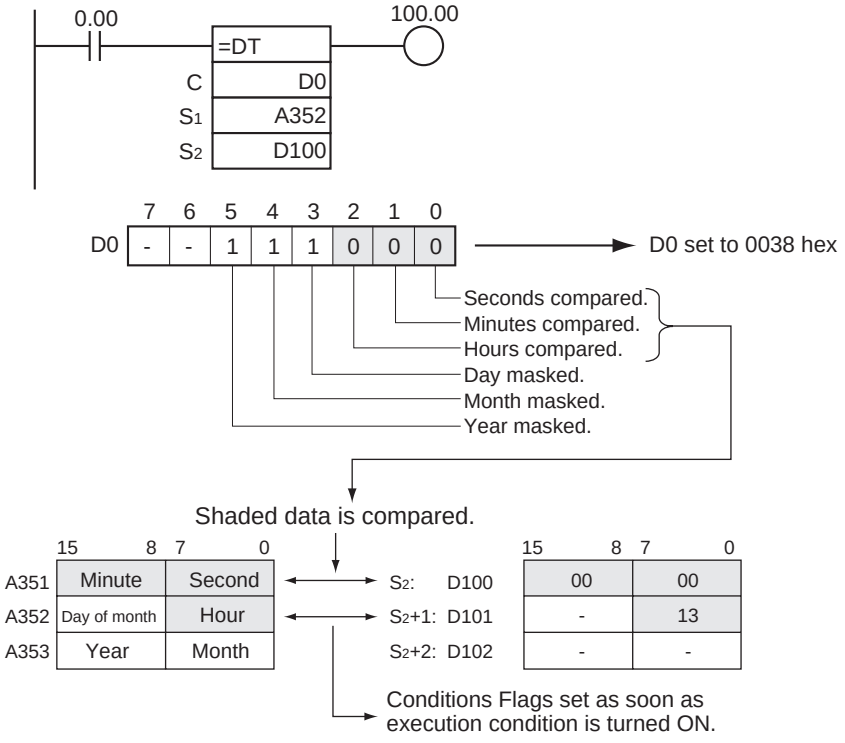
Name	Label	Operation
Error Flag	ER	ON if all 6 of the mask bits (C bits 00 to 05) are ON. OFF in all other cases.
Greater Than Flag	>	ON if $S_1 > S_2$. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if $S_1 \geq S_2$. OFF in all other cases.
Equal Flag	=	ON if $S_1 = S_2$. OFF in all other 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

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

Example

When CIO 0.00 is ON and the time is 13:00:00, CIO 100.00 is turned ON. The contents of A351 to A353 (the CPU Unit's internal calendar/clock data) are used as the present time data and the contents of D100 to D102 are used as the comparison time data. The year, month, and day values are masked, so only the hour, minute, and second data are compared.

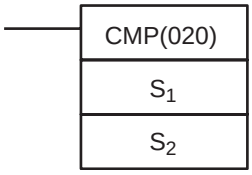


3-6-3 COMPARE: CMP(020)

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
Immediate Refreshing Specification		!CMP(020)

Applicable Program Areas

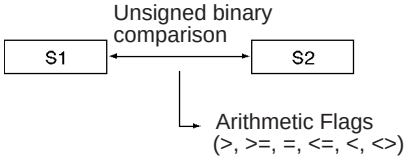
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal)	
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

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

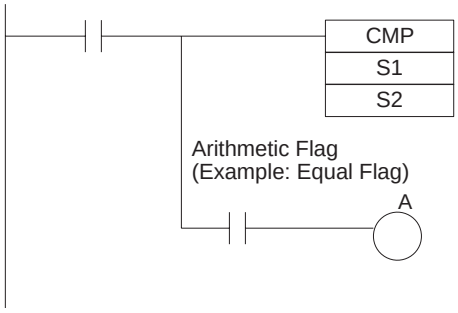
The following table shows the status of the Arithmetic Flags after execution of CMP(020). (A status of "---" indicates that the Flag may be ON or OFF.)

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 $S_1 = S_2$.

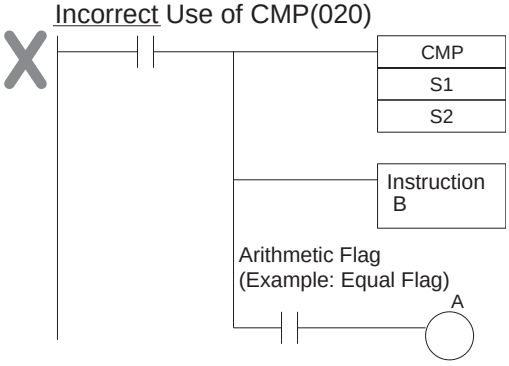
Correct Use of CMP(020)



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).

Incorrect Use of CMP(020)



The immediate-refreshing variation (!CMP(020)) can be used with words allocated to external inputs specified in S_1 and/or S_2. When !CMP(020) is executed, input refreshing will be performed for the external input word specified in S_1 and/or S_2 and that refreshed value will be compared.

Flags

Name	CX-Programmer 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

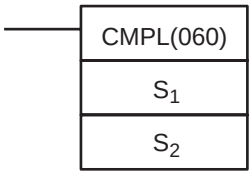
Do not program another instruction between CMP(020) and an input condition that accesses the result of CMP(020) because the other instruction might change the status of the Arithmetic Flags.

3-6-4 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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

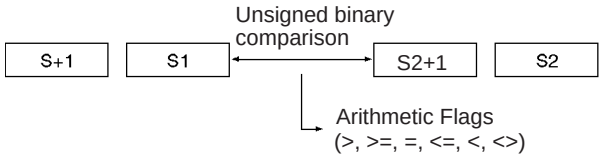
Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	

Area	S_1	S_2
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal)	
Data Registers	---	
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 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

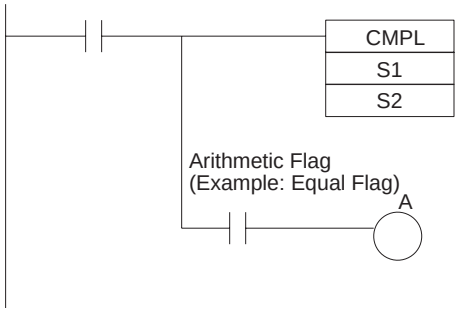
The following table shows the status of the Arithmetic Flags after execution of CMPL(060). (A status of "---" indicates that the Flag may be ON or OFF.)

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

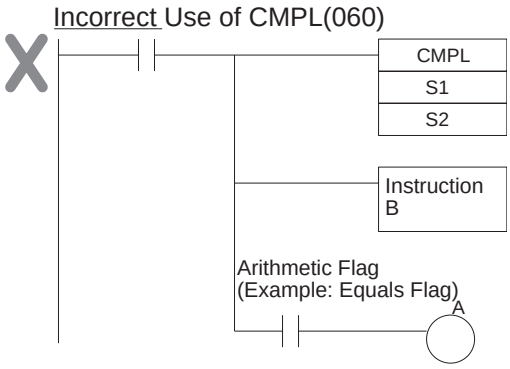
When CMPL(060) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CMPL(060), as shown in the following diagram. Here, the Equals Flag and output A will be turned ON when S_1 +1, S_1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



Flags

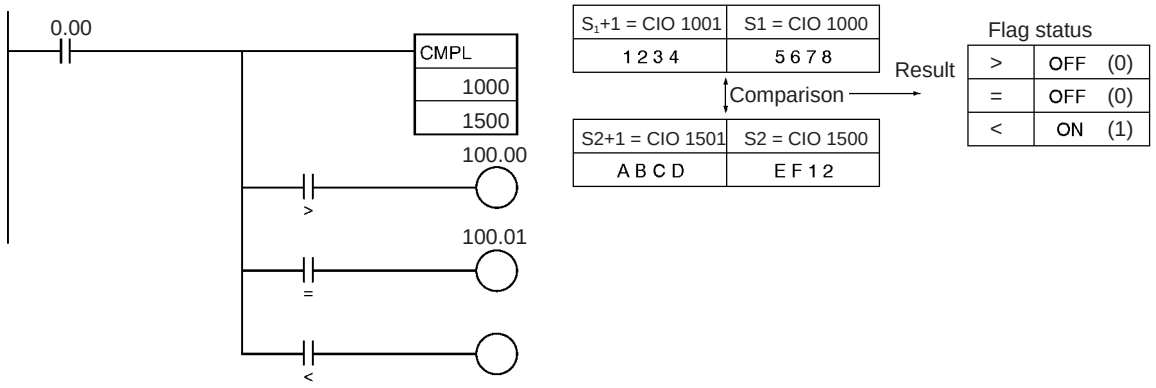
Name	CX-Programmer label	Operation
Greater Than Flag	P_GT	ON if S_1 +1, $S_1 > S_2$+1, S_2. OFF in all other cases.
Greater Than or Equal Flag	P_GE	ON if S_1 +1, $S_1 \geq S_2$+1, S_2. OFF in all other cases.
Equal Flag	P_EQ	ON if S_1 +1, $S_1 = S_2$+1, S_2. OFF in all other cases.
Not Equal Flag	P_NE	ON if S_1 +1, $S_1 \neq S_2$+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, $S_1 \leq S_2$+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 0.00 is ON in the following example, the eight-digit unsigned binary data in CIO 1001 and CIO 1000 is compared to the eight-digit unsigned binary data in CIO 1501 and CIO 1500 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 100.00 (Greater Than), CIO 100.01 (Equals), and CIO 100.02 (Less Than).

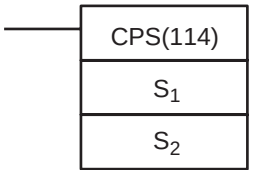


3-6-5 SIGNED BINARY COMPARE: CPS(114)

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
Immediate Refreshing Specification		!CPS(114)

Applicable Program Areas

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

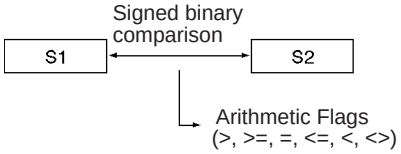
Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	

Area	S_1	S_2
Constants	#0000 to #FFFF (binary) –32768 to 0 to 32767 (signed decimal)	
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

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

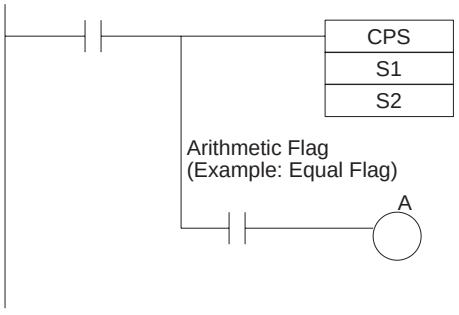
The following table shows the status of the Arithmetic Flags after execution of CPS(114). (A status of "---" indicates that the Flag may be ON or OFF.)

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 $S_1 = S_2$.

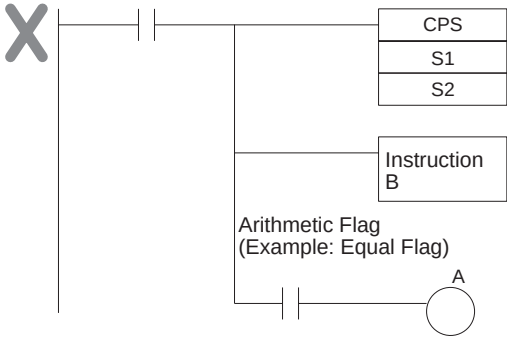
Correct Use of CPS(114)



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)



The immediate-refreshing variation (!CPS(114)) can be used with words allocated to external inputs specified in S_1 and/or S_2. When !CPS(114) is executed, input refreshing will be performed for the external input word specified in S_1 and/or S_2 and that refreshed value will be compared.

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

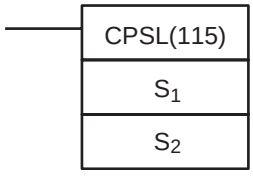
Do not program another instruction between CPS(114) and an input condition that accesses the result of CPS(114) because the other instruction might change the status of the Arithmetic Flags.

3-6-6 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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

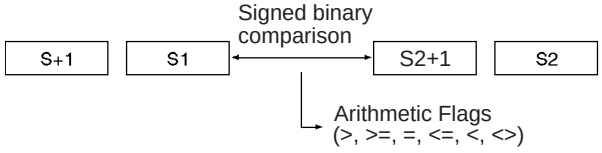
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary) –2147483648 to 0 to 2147483647 (signed decimal)	
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 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

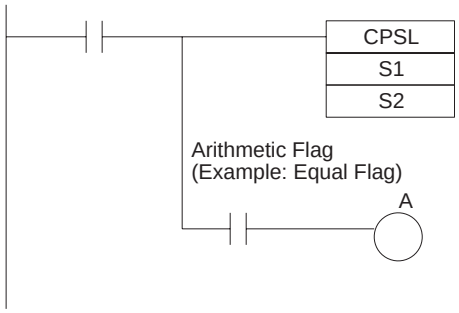
The following table shows the status of the Arithmetic Flags after execution of CPSL(115). (A status of "---" indicates that the Flag may be ON or OFF.)

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.

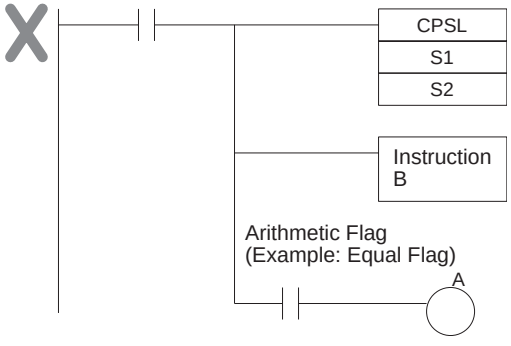
Correct Use of CPSL(115)



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).

Incorrect Use 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 ≥ S_2+1, S_2. OFF in all other cases.

Name	Label	Operation
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.

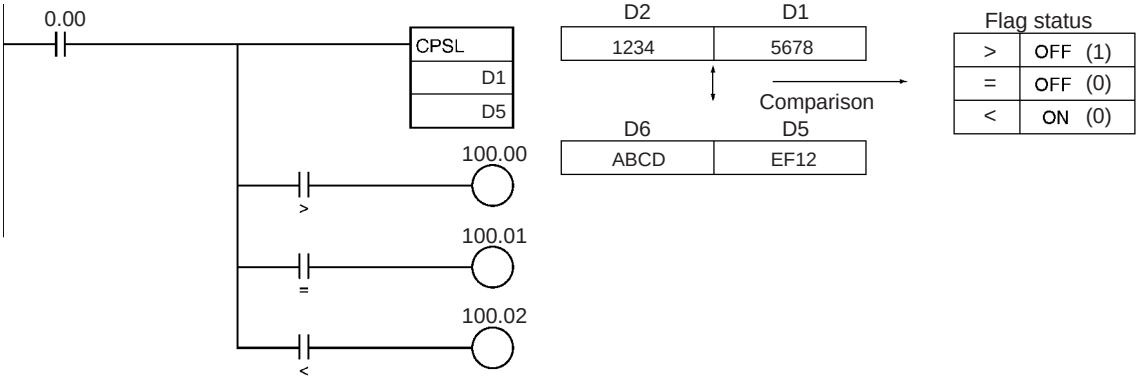
Precautions

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 0.00 is ON in the following example, the eight-digit signed binary data in D2 and D1 is compared to the eight-digit signed binary data in D6 and D5 and the result is output to the Arithmetic Flags.

- If the content of D2 and D1 is greater than that of D6 and D5, the Greater Than Flag will be turned ON, causing CIO 100.00 to be turned ON.
- If the content of D2 and D1 is equal to that of D6 and D5, the Equals Flag will be turned ON, causing CIO 100.01 to be turned ON.
- If the content of D2 and D1 is less than that of D6 and D5, the Less Than Flag will be turned ON, causing CIO 100.02 to be turned ON.

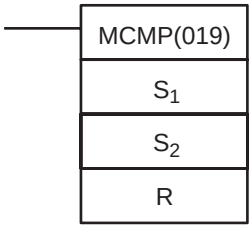


3-6-7 MULTIPLE COMPARE: MCMP(019)

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



S1: First word of set 1

S2: First word of set 2

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	MCMP(019)
	Executed Once for Upward Differentiation	@MCMP(019)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		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

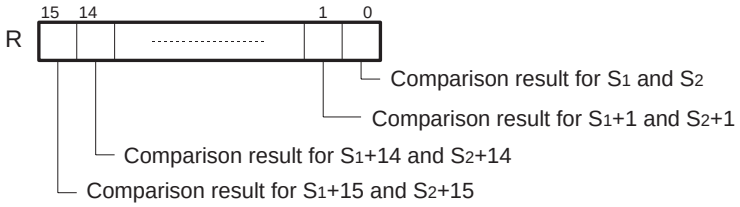
Specifies the beginning of the first 16-word range. S_1 and S_1+15 must be in the same data area.

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.



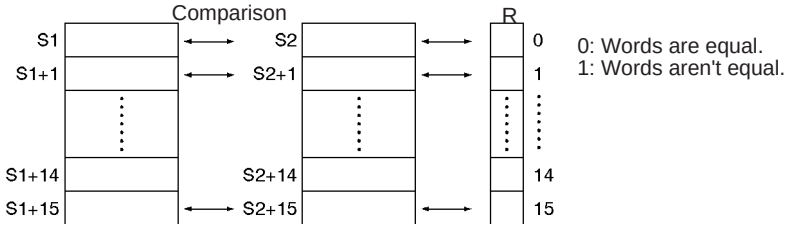
Operand Specifications

Area	S_1	S_2	R
CIO Area	CIO 0 to CIO 6128		CIO 0 to CIO 6143
Work Area	W0 to W496		W0 to W511
Holding Bit Area	H0 to H496		H0 to H511
Auxiliary Bit Area	A0 to A944		A448 to A959
Timer Area	T0000 to T4080		T0000 to T4095
Counter Area	C0000 to C4080		C0000 to C4095
DM Area	D0 to D32752		D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.

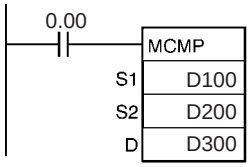


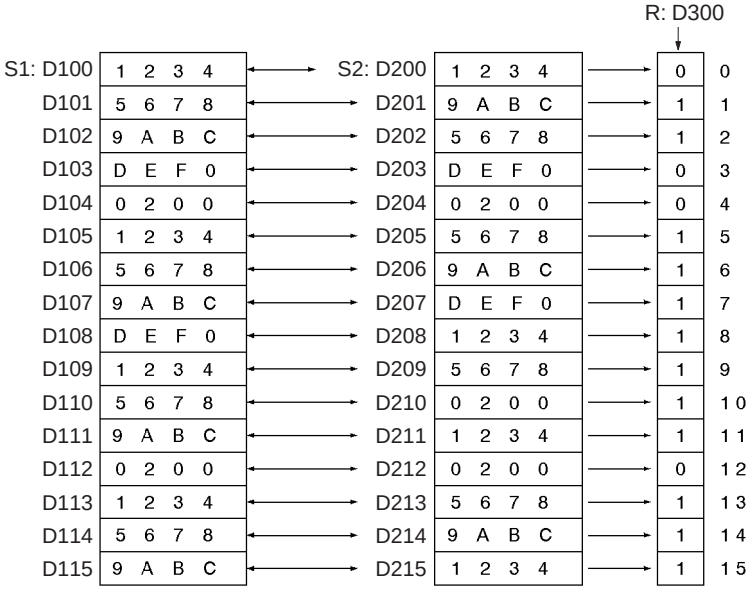
Flags

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 0.00 is ON in the following example, MCMP(019) compares words D100 through D115 in order to words D200 through D215 and turns ON the corresponding bits in D300 when the words **are not** equal.



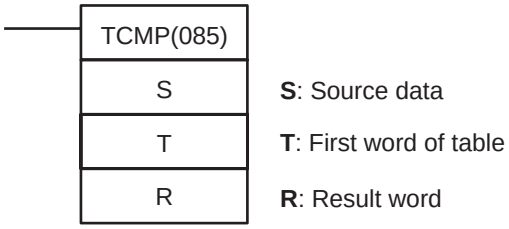


3-6-8 TABLE COMPARE: TCMP(085)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

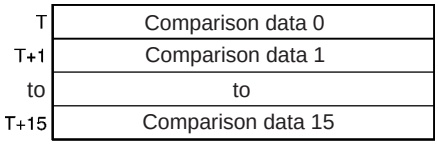
Operands

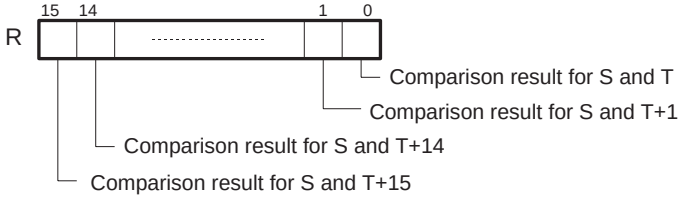
T: First word of table

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

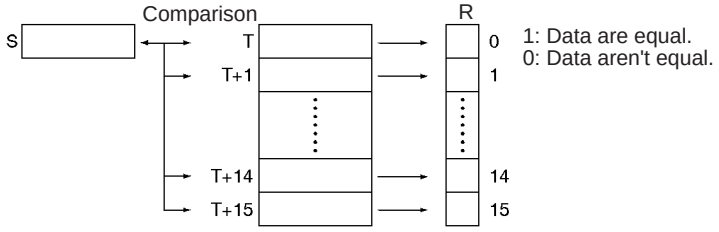
Area	S	T	R
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6128	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W496	W0 to W511
Holding Bit Area	H0 to H511	H0 to H496	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A944	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4080	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4080	C0000 to C4095
DM Area	D0 to D32767	D0 to D32752	D0 to D32767

Area	S	T	R
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)	---	
Data Registers	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 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.

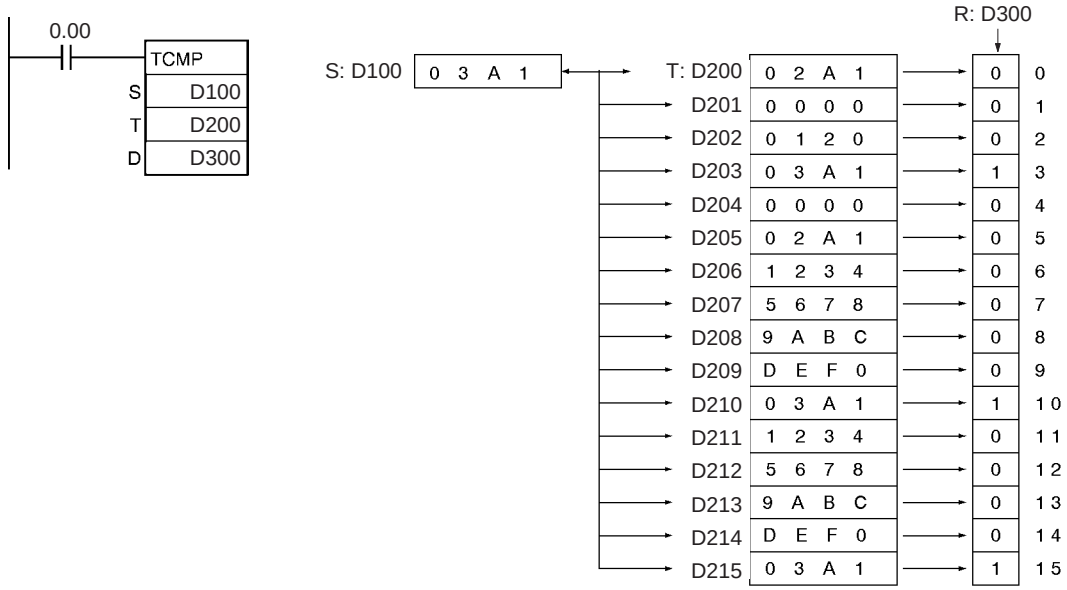


Flags

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 0.00 is ON in the following example, TCMP(085) compares the content of D100 with the contents of words D200 through D215 and turns ON the corresponding bits in D300 when the contents are equal or OFF when the contents are not equal.

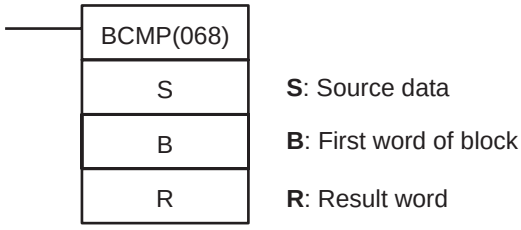


3-6-9 BLOCK COMPARE: BCMP(068)

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



Variations

Variations	**Executed Each Cycle for ON Condition**	BCMP(068)
	Executed Once for Upward Differentiation	@BCMP(068)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

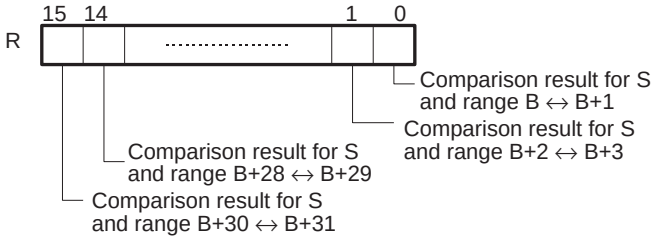
Operands

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 0 to CIO 6143	CIO 0 to CIO 6112	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W480	W0 to W511
Holding Bit Area	H0 to H511	H0 to H480	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A928	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4064	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4064	C0000 to C4095
DM Area	D0 to D32767	D0 to D32736	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)	---	
Data Registers	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 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

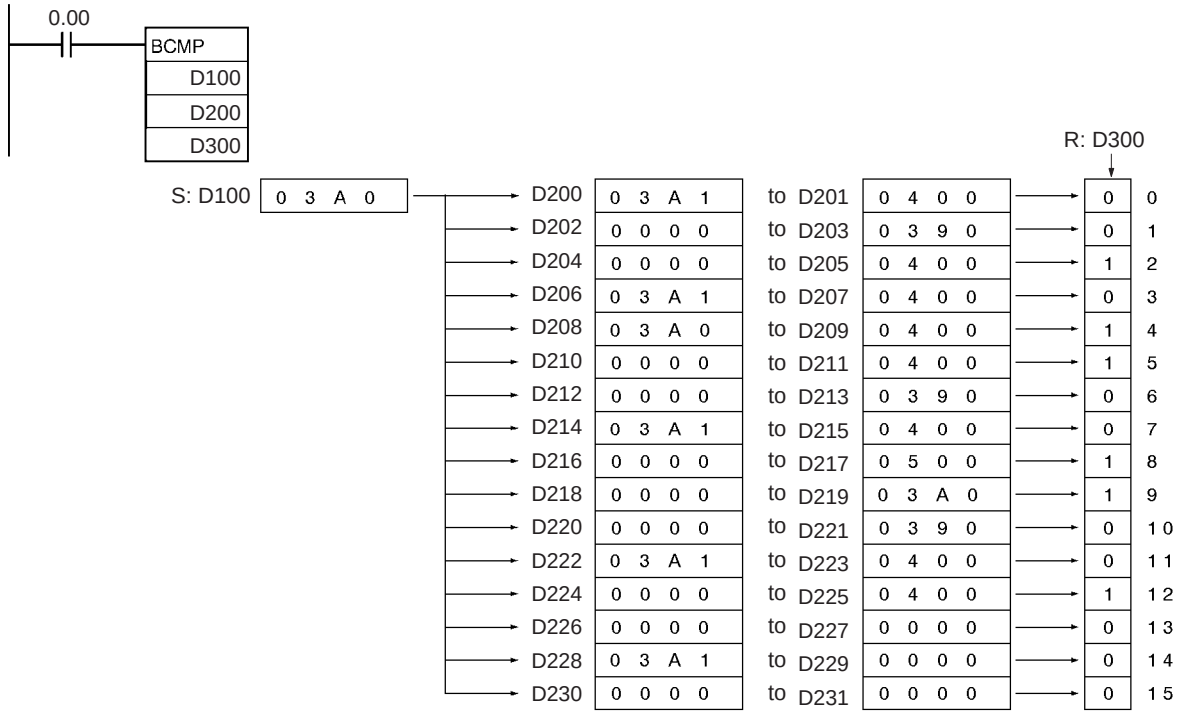
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (S is not within any of the 16 ranges.) OFF in all other cases.

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 0.00 is ON in the following example, BCMP(068) compares the content of D100 with the 16 ranges defined in D200 through D231 and turns ON the corresponding bits in D300 when S is within the range or OFF when S is not within the range.

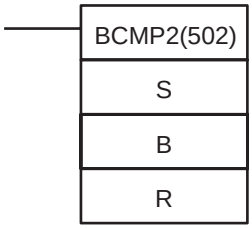


3-6-10 EXPANDED BLOCK COMPARE: BCMP2(502)

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
Immediate Refreshing Specification		Not supported

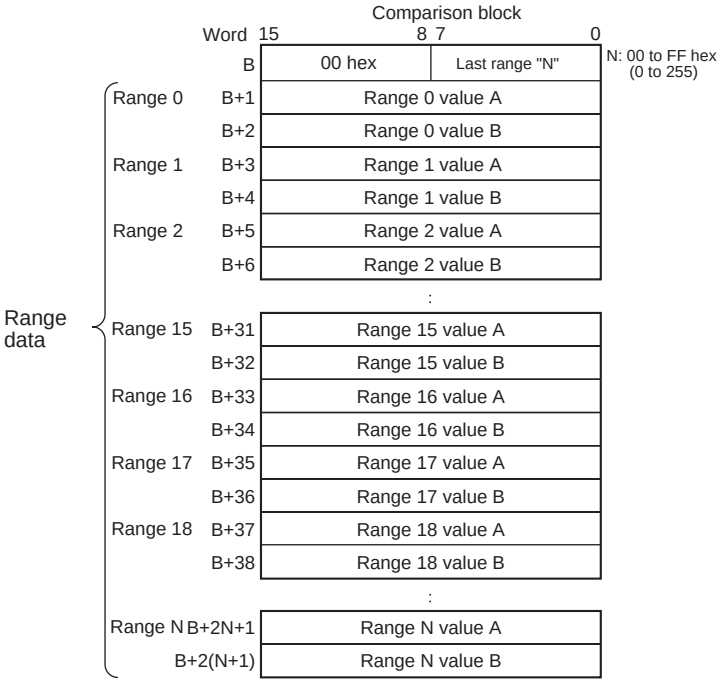
Applicable Program Areas

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

Operands

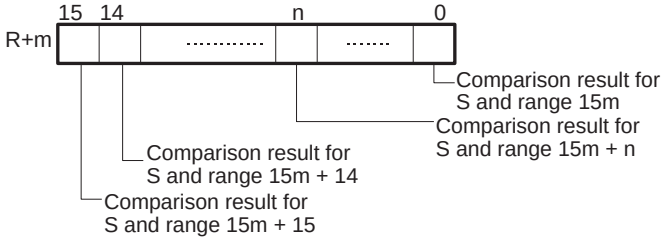
B: First word of block

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 the comparison block. The maximum number of result words is 16, i.e., m equals 0 to 15.



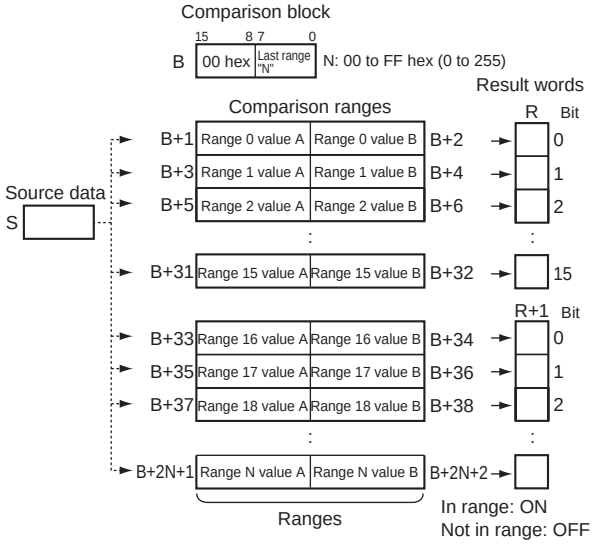
Operand Specifications

Area	S	B	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 32767 (signed decimal)	---	
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

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.



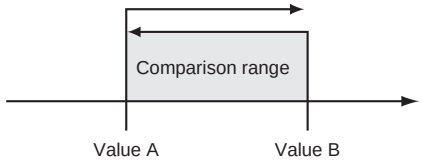
Number of Ranges

The number of ranges in the comparison block is set in the first word of the block. Up to 256 ranges can be set.

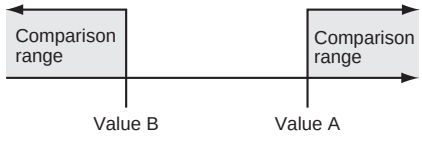
Setting Ranges

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



· If Value A > Value B
Then, Comparison range ≤ Value B and Value A ≤ Comparison range



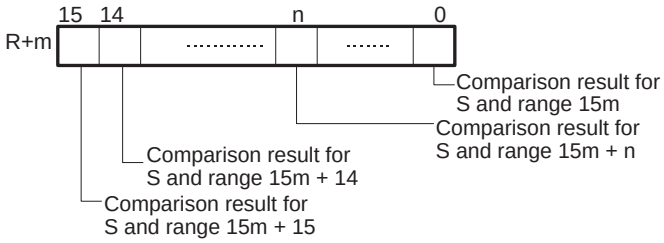
Example

When B+1 ≤ B+2
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.

When B+1 > B+2
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.

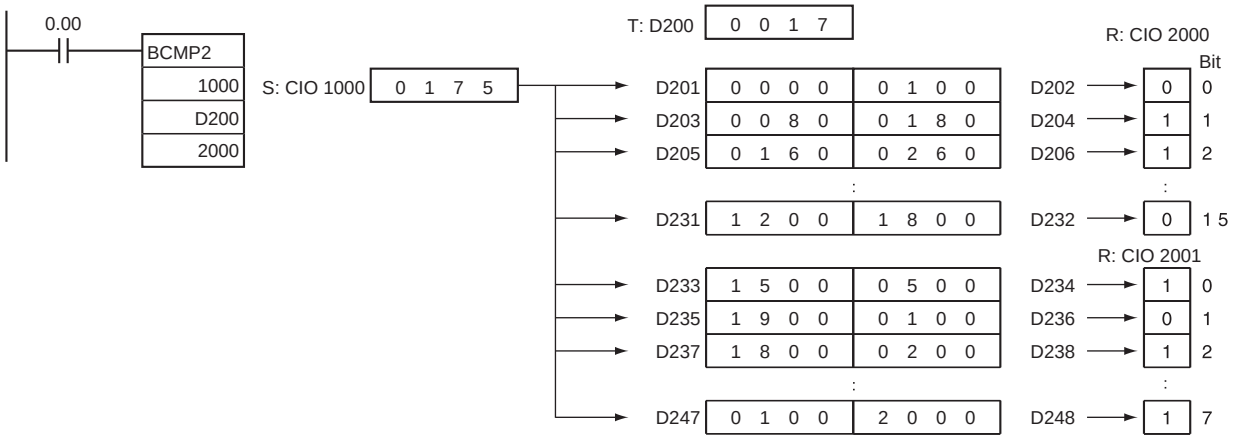


Flags

Name	Label	Operation
Error Flag	ER	OFF

Example

When CIO 0.00 is ON in the following example, BCMP2(502) compares the content of CIO 1000 with the 24 ranges defined in D200 through D247 (N = 17 hex = 23 decimal, i.e., 24 ranges) and turns ON the corresponding bits in CIO 2000 and CIO 2001 when S is within the range and OFF when S is not within the range. For example, if the source data in CIO 1000 is in the range defined by D201 and D202, then bit 00 of CIO 2000 is turned ON and if it in not in the range, then bit 00 of CIO 2000 is turned OFF. Likewise, the source data in CIO 1000 is compared to the ranges defined by D203 and D204, D247 and D248, and the other words in the comparison block, and bit 1 in CIO 2000, bit 7 in CIO 2001, and the other bits in the result words are manipulated according to the results of comparison.

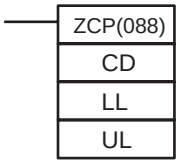


3-6-11 AREA RANGE COMPARE: ZCP(088)

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
Immediate Refreshing Specification		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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)		
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

ZCP(088) compares the 16-bit signed 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

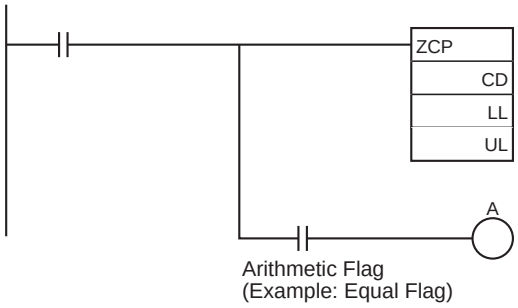
The following table shows the status of the Arithmetic Flags after execution of ZCP(088).

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

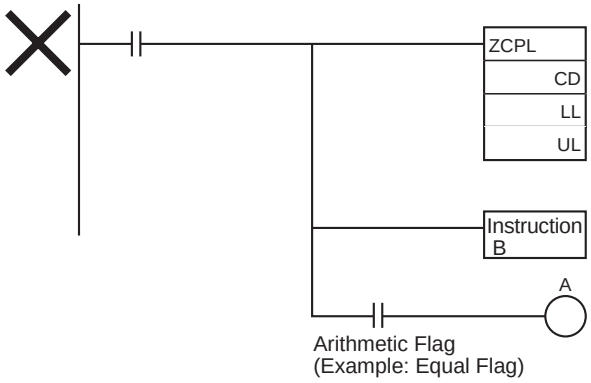
When ZCP(088) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls ZCP(088), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when LL ≤ 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.

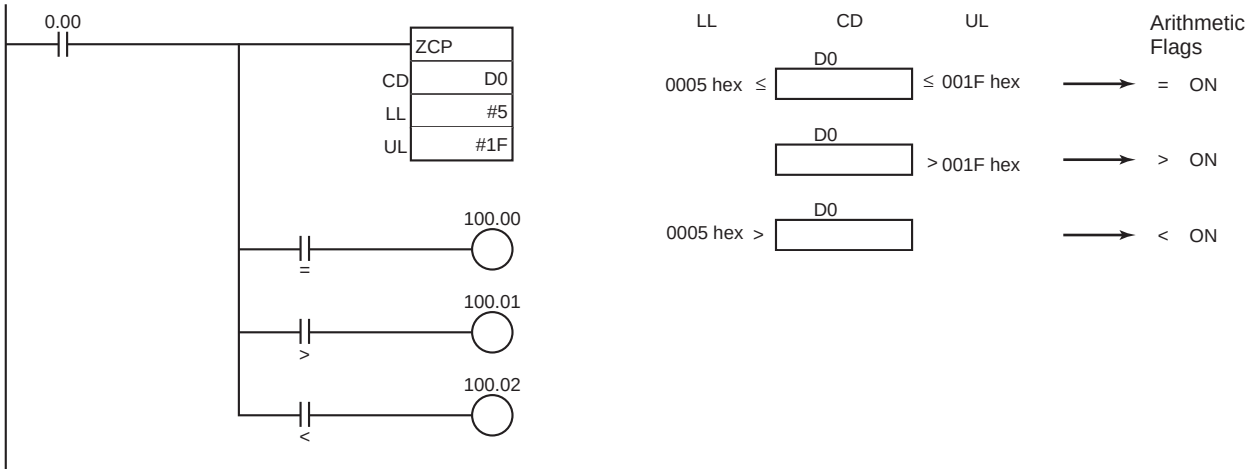
Name	Label	Operation
Equal Flag	=	ON if LL ≤ CD ≤ UL. OFF in all other cases.
Not Equal Flag	<>	Left unchanged.
Less Than Flag	<	ON if CD < LL. OFF in all other cases.
Less Than or Equal Flag	< =	Left unchanged.
Negative Flag	N	Left unchanged.

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 0.00 is ON in the following example, the 16-bit unsigned binary data in D0 is compared to the range 0005 to 001F hex (5 to 31 decimal) and the result is output to the Arithmetic Flags.
CIO 100.00 is turned ON if 0005 hex ≤ content of D0 ≤ 001F hex.
CIO 100.01 is turned ON if the content of D0 > 001F hex.
CIO 100.02 is turned ON if the content of D0 < 0005 hex.

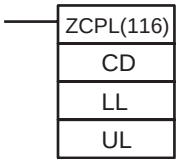


3-6-12 DOUBLE AREA RANGE COMPARE: ZCPL(116)

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
Immediate Refreshing Specification		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 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary) &0 to &4294967295 (unsigned decimal)		
Data Registers	---		
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ZCPL(116) compares the 32-bit signed 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 *3-6-11 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 > 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.
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.

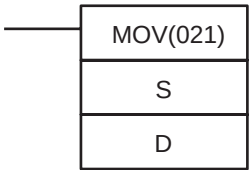
Instruction	Mnemonic	Function code	Page
MOVE	MOV	021	248
MOVE NOT	MVN	022	249
DOUBLE MOVE	MOVL	498	251
DOUBLE MOVE NOT	MVNL	499	252
MOVE BIT	MOVB	082	254
MOVE DIGIT	MOVD	083	256
MULTIPLE BIT TRANSFER	XFRB	062	258
BLOCK TRANSFER	XFER	070	261
BLOCK SET	BSET	071	263
DATA EXCHANGE	XCHG	073	265
DOUBLE DATA EXCHANGE	XCGL	562	266
SINGLE WORD DISTRIBUTE	DIST	080	268
DATA COLLECT	COLL	081	270
MOVE TO REGISTER	MOVR	560	271
MOVE TIMER/COUNTER PV TO REGISTER	MOVRW	561	273

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
Immediate Refreshing Specification		!MOV(021)
Combined Variations	**Executed Once and Destination Refreshed Immediately for Upward Differentiation**	!@MOV(021)

Applicable Program Areas

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

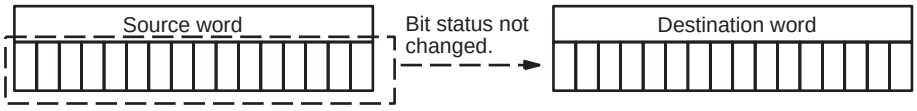
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	

Area	S	D
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
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 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.



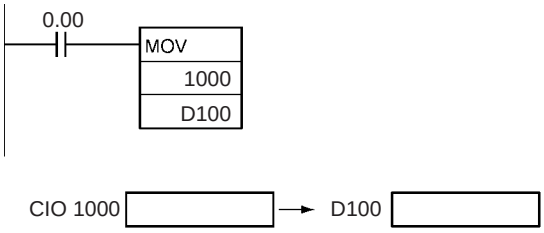
MOV(021) has an immediate refreshing variation (!MOV(021)). External input bits can be specified for S and external output bits can be specified for D. Input bits used for S will refreshed just before, and output bits used for D will be refreshed just after execution.

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 0.00 is ON in the following example, the content of CIO 1000 is copied to D100.

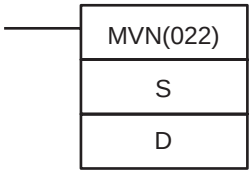


3-7-2 MOVE NOT: MVN(022)

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
Immediate Refreshing Specification		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 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

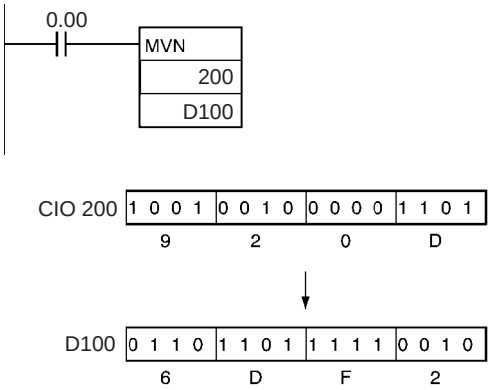
MVN(022) inverts the bits in S and transfers the result to D. The content of S is left unchanged.

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 0.00 is ON in the following example, the status of the bits in CIO 200 is inverted and the result is copied to D100.

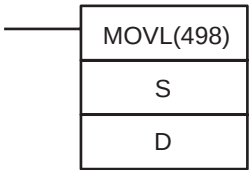


3-7-3 DOUBLE MOVE: MOVL(498)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

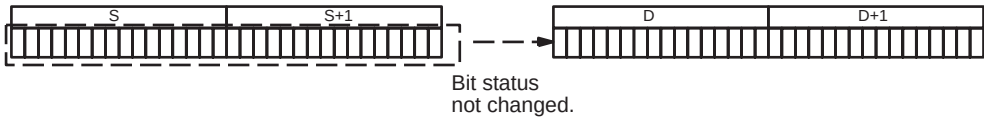
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	

Area	S	D
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, 1–(– –) IR5	

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.

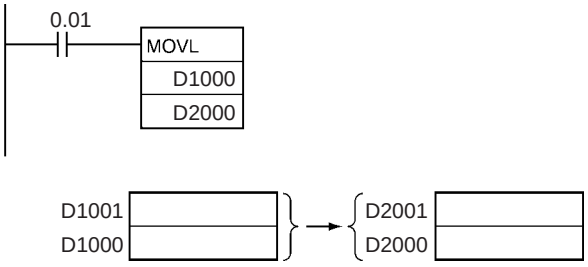


Flags

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 0.01 is ON in the following example, the content of D1001 and D1000 are copied to D2001 and D2000.

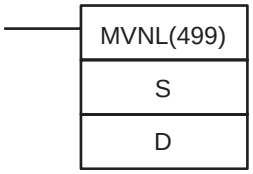


3-7-4 DOUBLE MOVE NOT: MVNL(499)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

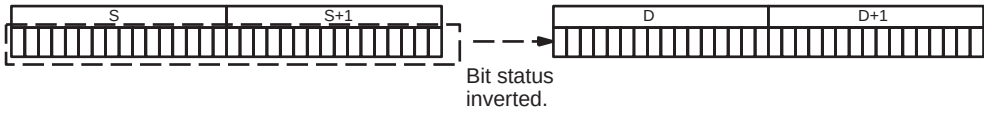
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.

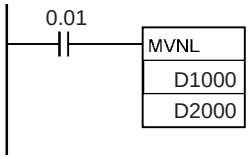


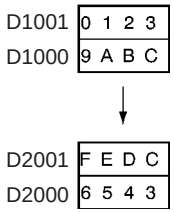
Flags

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 0.01 is ON in the following example, the status of the bits in D1001 and D1000 are inverted and the result is copied to D2001 and D2000. (The original contents of D1001 and D1000 are left unchanged.)



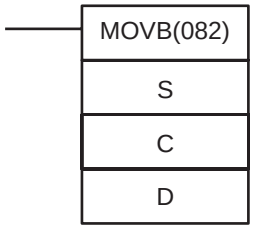


3-7-5 MOVE BIT: MOVB(082)

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
Immediate Refreshing Specification		Not supported

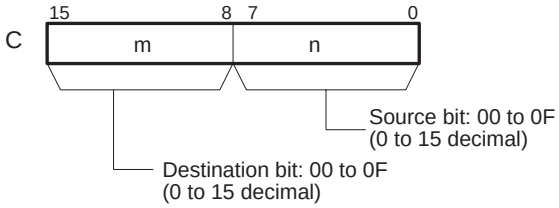
Applicable Program Areas

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.



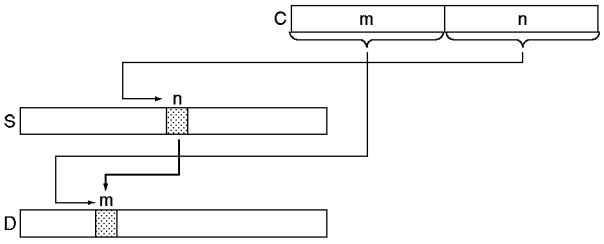
Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		

Area	S	C	D
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	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

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.



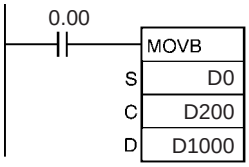
Note The same word can be specified for both S and D to copy a bit within a word.

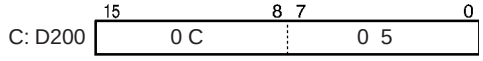
Flags

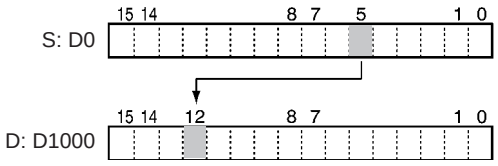
Name	Label	Operation
Error Flag	ER	ON if the rightmost and leftmost two digits of C are not within the specified range of 00 to 0F. OFF in all other cases.

Examples

When CIO 0.00 is ON in the following example, the 5th bit of the source word (D0) is copied to the 12th bit of the destination word (D1000) in accordance with the control word's value of 0C05.





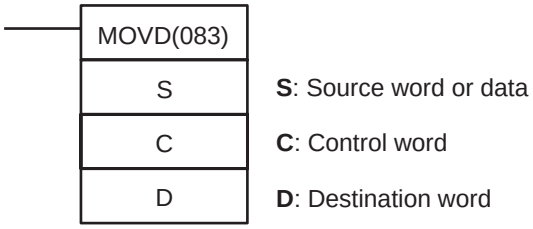


3-7-6 MOVE DIGIT: MOVD(083)

Purpose

Transfers the specified digit or digits. (Each digit is made up of 4 bits.)

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MOVD(083)
	Executed Once for Upward Differentiation	@MOVD(083)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

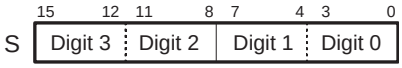
Applicable Program Areas

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

Operands

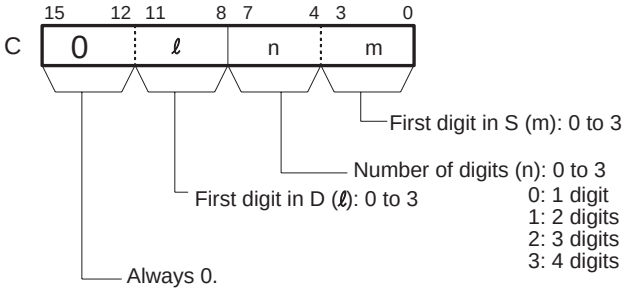
S: Source Word

The source digits are read from right to left, wrapping back to the rightmost digit (digit 0) if necessary.



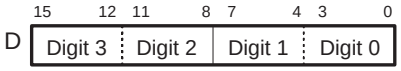
C: Control Word

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

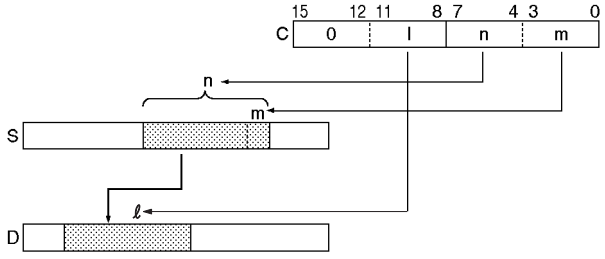
Area	S	C	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		

Area	S	C	D
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	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

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.



Note The same word can be specified for both S and D to copy a bit within a word.

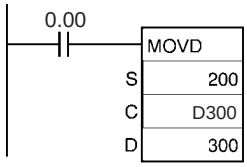
Flags

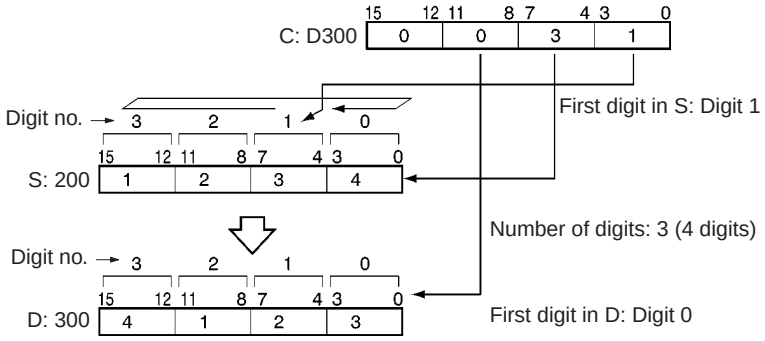
Name	Label	Operation
Error Flag	ER	ON if one of the first three digits of C is not within the specified range of 0 to 3. OFF in all other cases.

Examples

Four-digit Transfer

When CIO 0.00 is ON in the following example, four digits of data are copied from CIO 200 to CIO 300. The transfer begins with the digit 1 of CIO 200 and digit 0 or CIO 300, in accordance with the control word's value of 0031.

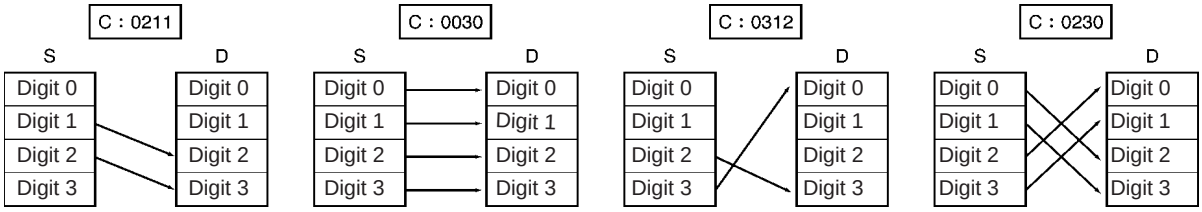




Note After reading the leftmost digit of S (digit 3), MOVD(083) wraps to the rightmost digit (digit 0).

Examples of C

The following diagram shows examples of data transfers for various values of C.

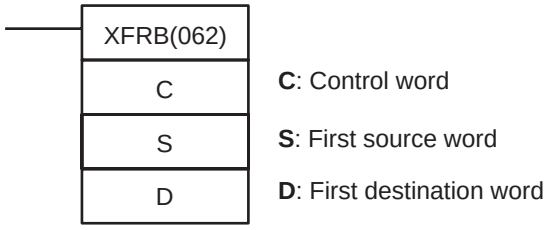


3-7-7 MULTIPLE BIT TRANSFER: XFRB(062)

Purpose

Transfers the specified number of consecutive bits.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	XFRB(062)
	Executed Once for Upward Differentiation	@XFRB(062)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

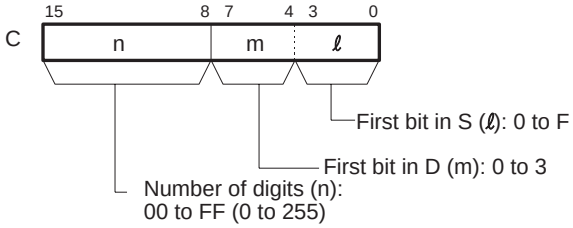
Applicable Program Areas

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

Operands

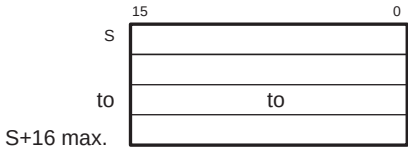
C: Control Word

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.



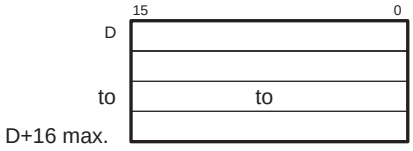
S: First Source Word

Specifies the first source word. Bits are read from right to left, continuing with consecutive words (up to S+16) when necessary.



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.



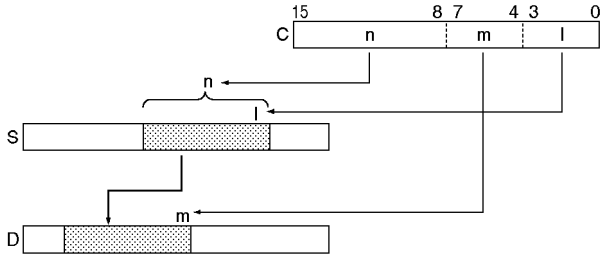
Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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

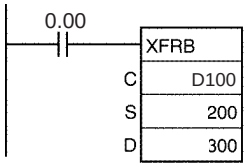
Precautions

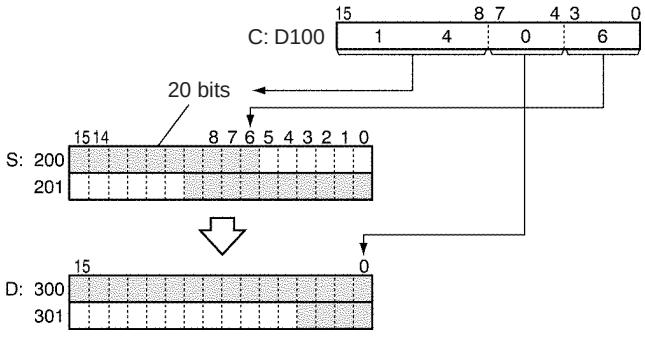
Up to 255 bits of data can be transferred per execution of XFRB(062).

Be sure that the source words and destination words do not exceed the end of the data area.

Examples

When CIO 0.00 is ON in the following example, the 20 bits beginning with CIO 200.06 are copied to the 20 bits beginning with CIO 300.00.



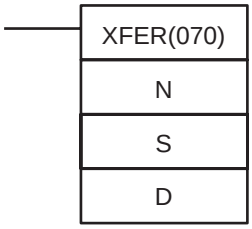


3-7-8 BLOCK TRANSFER: XFER(070)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

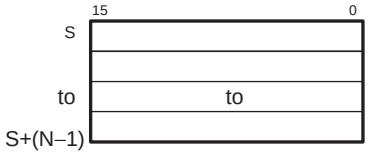
Operands

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).

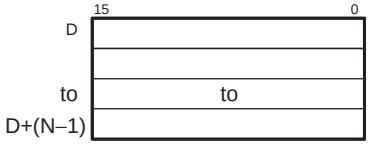
S: First Source Word

Specifies the first source word.



D: First Destination Word

Specifies the first destination word.



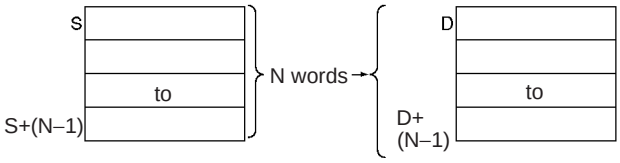
Operand Specifications

Area	N	S	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		

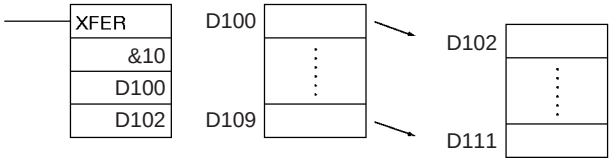
Area	N	S	D
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) or &0 to &65535	---	---
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) 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

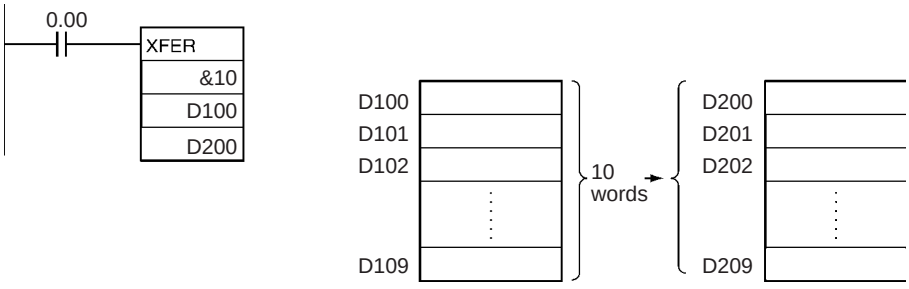
Precautions

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 0.00 is ON in the following example, the 10 words D100 through D109 are copied to D200 through D209.

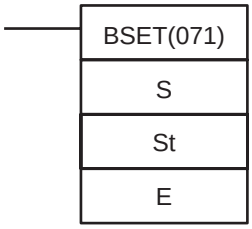


3-7-9 BLOCK SET: BSET(071)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: Source Word

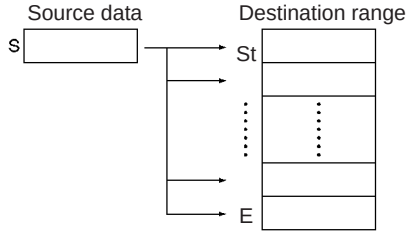
Specifies the source data or the word containing the source data.

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.

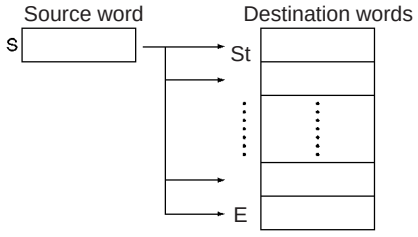
Operand Specifications

Area	S	St	E
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		

Area	S	St	E
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, 15–(– –) IR		

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.

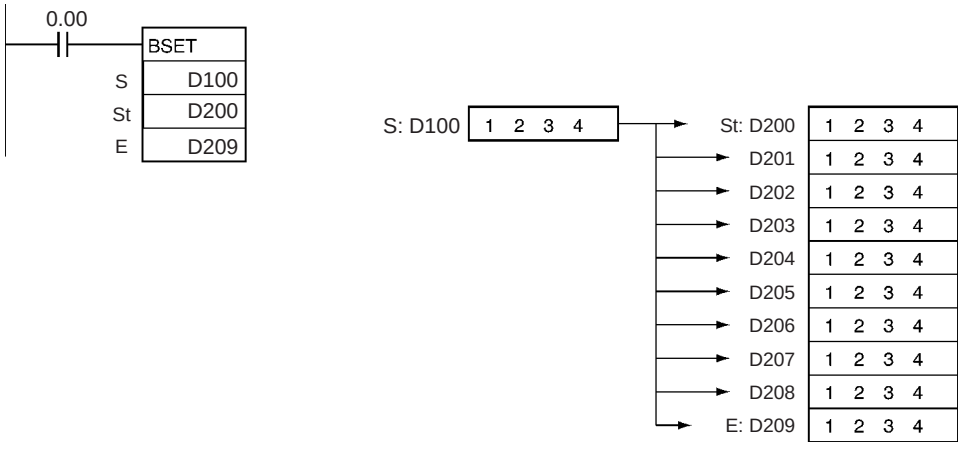
Precautions

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 0.00 is ON in the following example, the source data in D100 is copied to D200 through D209.

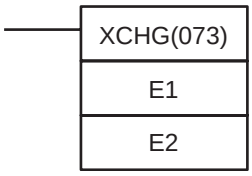


3-7-10 DATA EXCHANGE: XCHG(073)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

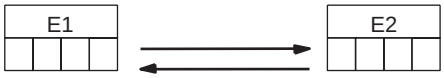
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	E1	E2
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

XCHG(073) exchanges the contents of E1 and E2.

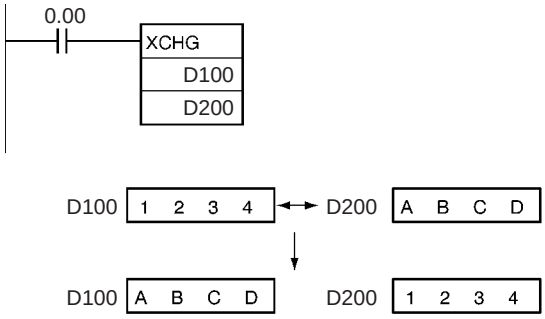


Flags

There are no flags affected by this instruction.

Example

When CIO 0.00 is ON in the following example, the content of D100 is exchanged with the content of D200.

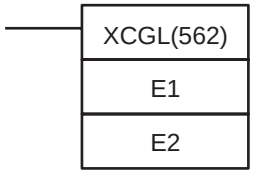


3-7-11 DOUBLE DATA EXCHANGE: XCGL(562)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

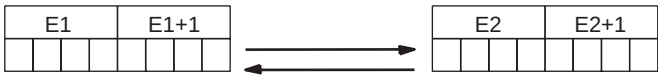
Operand Specifications

Area	E1	E2
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A448 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	---
Data Registers	---	

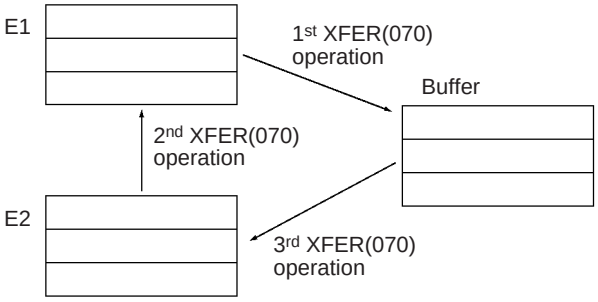
Area	E1	E2
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

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.

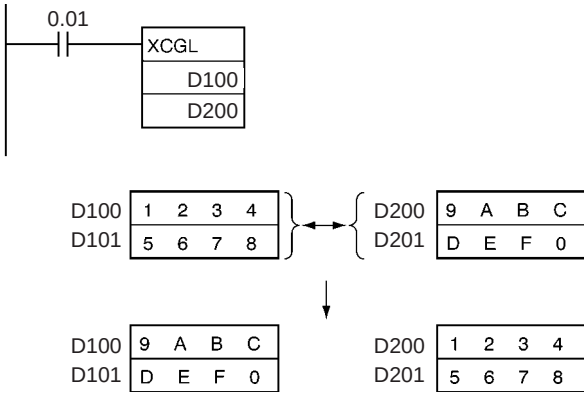


Flags

There are no flags affected by this instruction.

Example

When CIO 0.01 is ON in the following example, the contents of D100 and D101 are exchanged with the contents of D200 and D201.

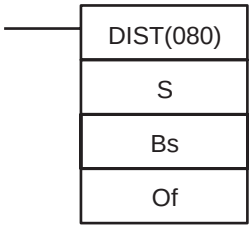


3-7-12 SINGLE WORD DISTRIBUTE: DIST(080)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

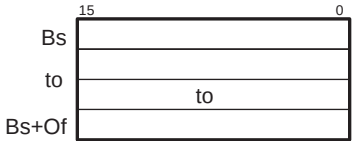
Operands

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.



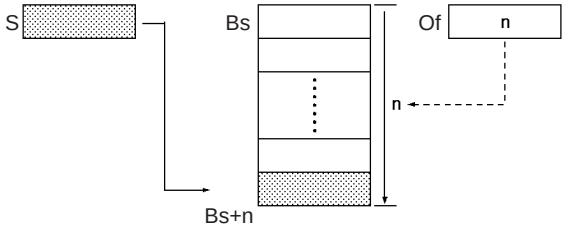
Operand Specifications

Area	S	Bs	Of
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	A0 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	---	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	DR0 to DR15	---	DR0 to DR15

Area	S	Bs	Of
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

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

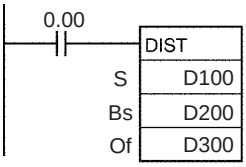
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the source data is 1. OFF in all other cases.

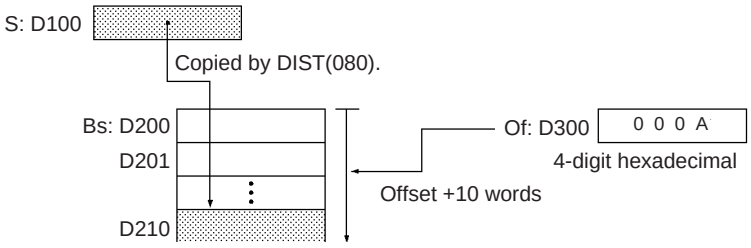
Precautions

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 0.00 is ON in the following example, the contents of D100 will be copied to D210 (D200 + 10) if the contents of D300 is 10 (0A hexadecimal). The contents of D100 can be copied to other words by changing the offset in D300.



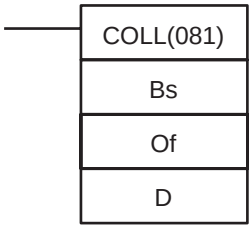


3-7-13 DATA COLLECT: COLL(081)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

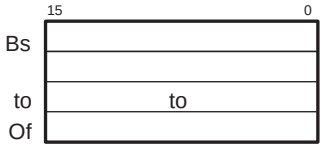
Operands

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.



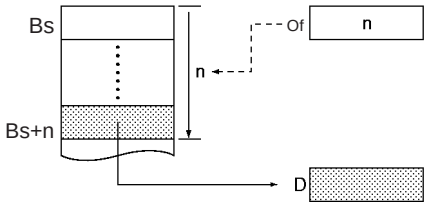
Operand Specifications

Area	Bs	Of	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0000 to #FFFF (binary) or &0 to &65535	---
Data Registers	---	DR0 to DR15	

Area	Bs	Of	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 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

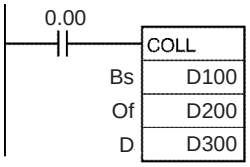
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the source data is 1. OFF in all other cases.

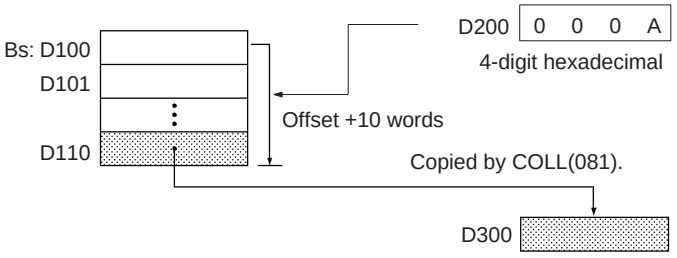
Precautions

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 0.00 is ON in the following example, the contents of D110 (D100 + 10) will be copied to D300 if the content of D200 is 10 (0A hexadecimal). The contents of other words can be copied to D300 by changing the offset in D200.



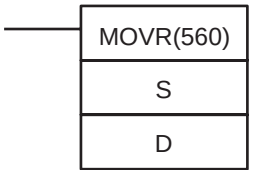


3-7-14 MOVE TO REGISTER: MOVR(560)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

D: Destination

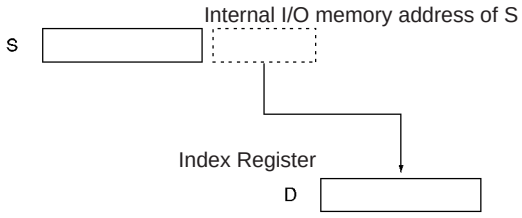
The destination must be an Index Register (IR0 to IR15).

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6143 CIO 0.00 to CIO 6143.15	---
Work Area	W0 to W511 W0.00 to W511.15	---
Holding Bit Area	H0 to H511 H0.00 to H511.15	---
Auxiliary Bit Area	A0 to A447 A448 to A959 A0.00 to A447.15 A448.00 to A959.15	---
Timer Area	T0000 to T4095 (Completion Flag)	---
Counter Area	C0000 to C4095 (Completion Flag)	---
Task Flag	TK00 to TK31	---
DM Area	D0 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

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

Precautions

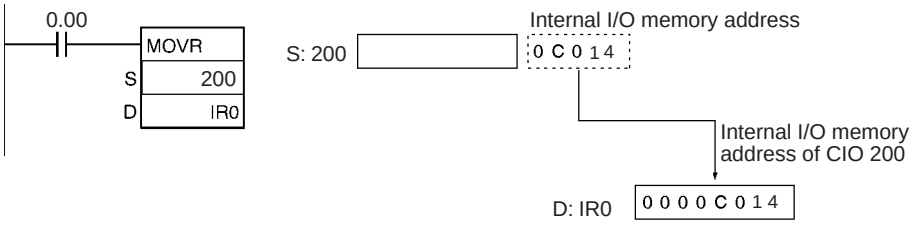
MOVR(560) cannot set the PLC memory addresses of timer/counter PVs. Use MOVRW(561) to set the PLC memory addresses of timer/counter PVs.

The contents of an index register in an interrupt task is not predictable until it is set. Be sure to set a register using MOVR(560) in an interrupt task before using the register.

Any changes to the contents of an IR or DR made in an interrupt task will not affect the contents of the register in a cyclic task.

Example

When CIO 0.00 is ON in the following example, MOVR(560) writes the PLC memory address of CIO 200 to IR0.

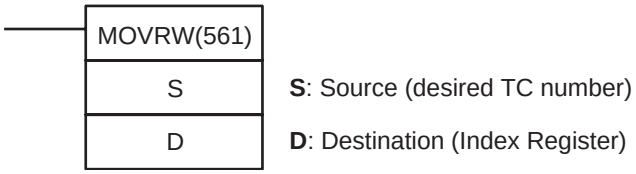


3-7-15 MOVE TIMER/COUNTER PV TO REGISTER: MOVRW(561)

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
Immediate Refreshing Specification		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).

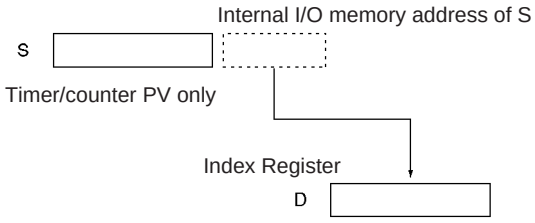
Operand Specifications

Area	S	D
CIO Area	---	
Work Area	---	
Holding Bit Area	---	

Area	S	D
Auxiliary Bit Area	---	
Timer Area	T0000 to T4095 (present value)	---
Counter Area	C0000 to C4095 (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

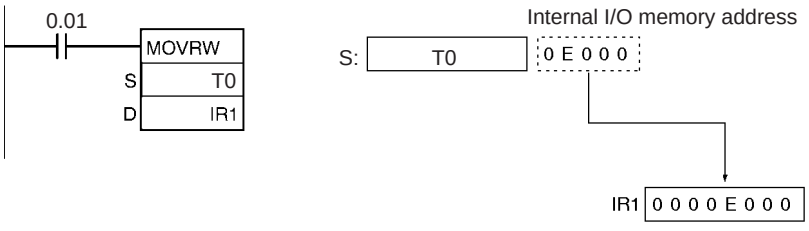
Name	Label	Operation
Error Flag	ER	OFF or unchanged
Equals Flag	=	OFF or unchanged
Negative Flag	N	OFF or unchanged

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 0.01 is ON in the following example, MOVRW(561) writes the PLC memory address for the PV of timer T0 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.

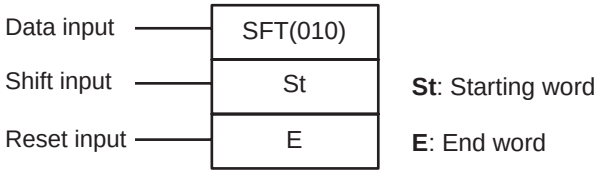
Instruction	Mnemonic	Function code	Page
SHIFT REGISTER	SFT	010	275
REVERSIBLE SHIFT REGISTER	SFTR	084	277
ASYNCHRONOUS SHIFT REGISTER	ASFT	017	280
WORD SHIFT	WSFT	016	282
ARITHMETIC SHIFT LEFT	ASL	025	284
DOUBLE SHIFT LEFT	ASLL	570	285
ARITHMETIC SHIFT RIGHT	ASR	026	287
DOUBLE SHIFT RIGHT	ASRL	571	288
ROTATE LEFT	ROL	027	290
DOUBLE ROTATE LEFT	ROLL	572	291
ROTATE LEFT WITHOUT CARRY	RLNC	574	296
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	298
ROTATE RIGHT	ROR	028	293
DOUBLE ROTATE RIGHT	RORL	573	295
ROTATE RIGHT WITHOUT CARRY	RRNC	575	300
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	301
ONE DIGIT SHIFT LEFT	SLD	074	303
ONE DIGIT SHIFT RIGHT	SRD	075	304
SHIFT N-BIT DATA LEFT	NSFL	578	306
SHIFT N-BIT DATA RIGHT	NSFR	579	308
SHIFT N-BITS LEFT	NASL	580	310
DOUBLE SHIFT N-BITS LEFT	NSLL	582	312
SHIFT N-BITS RIGHT	NASR	581	315
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	318

3-8-1 SHIFT REGISTER: SFT(010)

Purpose

Operates a shift register.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SFT(010)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

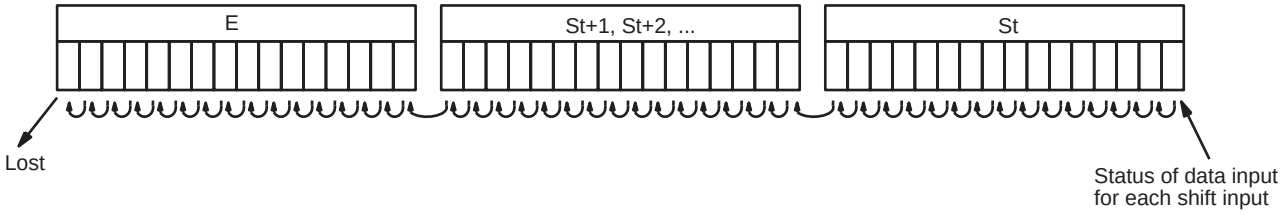
Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	---	
Counter Area	---	
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 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to 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, HR, 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.

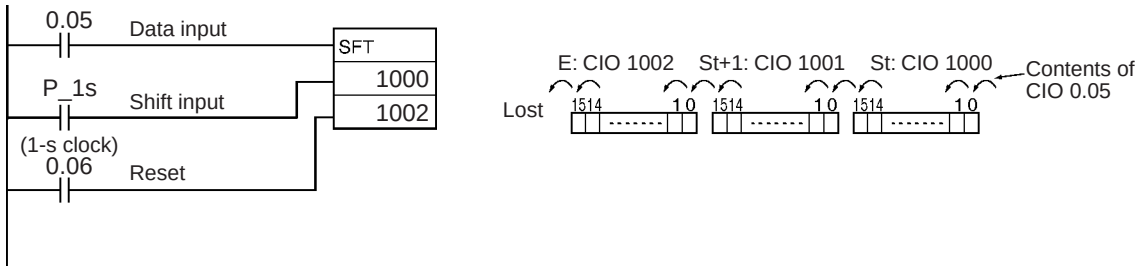
St must be less than or equal to E, but even when St is set to greater than E an error will not occur and one word of data in St will be shifted.

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 1000 to CIO 1002. A 1-s clock pulse is used so that the execution condition produced by CIO 0.05 is shifted into a 3-word register between CIO 1000.00 and CIO 1002.15 every second.

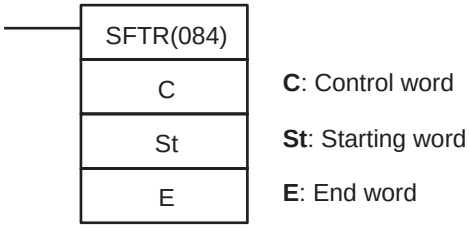


3-8-2 REVERSIBLE SHIFT REGISTER: SFTR(084)

Purpose

Creates a shift register that shifts data to either the right or the left.

Ladder Symbol



Variations

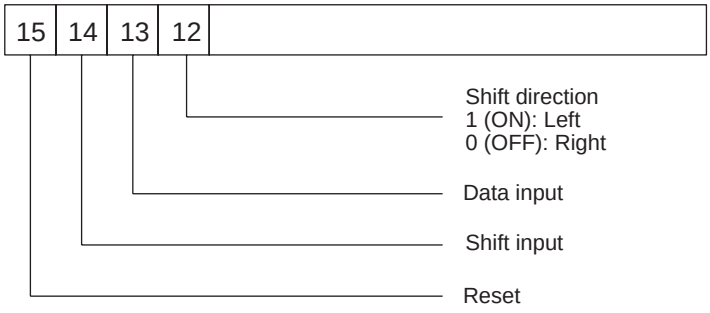
Variations		
	Executed Each Cycle for ON Condition	SFTR(084)
	Executed Once for Upward Differentiation	@SFTR(084)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

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.

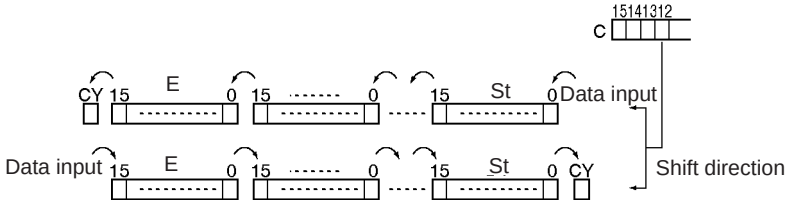
Operand Specifications

Area	C	St	E
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		

Area	C	St	E
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.

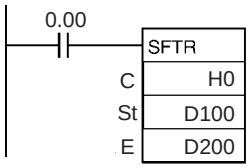
When reset (bit 15 of C) turns ON all bits in the shift register, from St to E will be reset (i.e., set to 0).

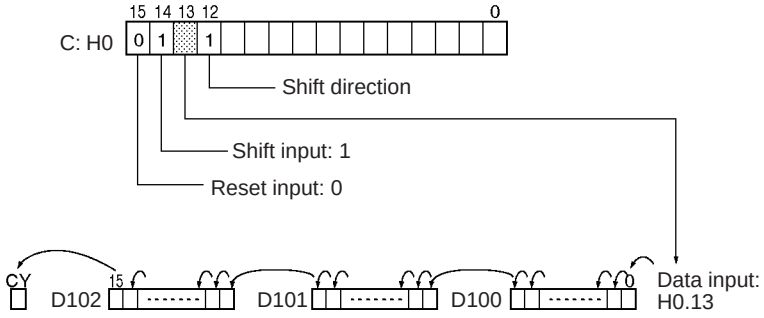
When St is greater than E, an error will be generated and the Error Flag will turn ON.

Examples

Shifting Data

If shift input H0.14 goes ON when CIO 0.00 is ON and the reset bit H0.15 is OFF, words D100 through D102 will shift one bit in the direction designated by H0.12 (e.g., 1: right) and the contents of input bit H0.13 will be shifted into the rightmost bit of D100. The contents of bit 15 of D102 will be shifted to the Carry Flag (CY).





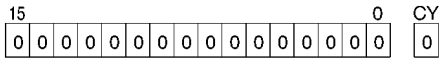
Resetting Data

If H0.14 is ON when CIO 0.00 is ON, and the reset bit, H0.15, is ON, words D100 through D102 and the Carry Flag will be reset to OFF.

Controlling Data

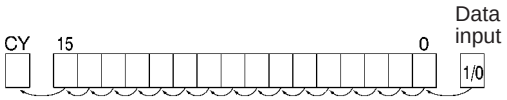
Resetting Data

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.



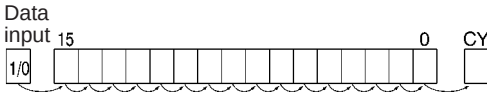
Shifting Data Left (from Rightmost to Leftmost Bit)

When the shift input bit (bit 14 of C) is ON, the contents of the input bit (bit 13 of C) is shifted 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) (I/O) is shifted 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.

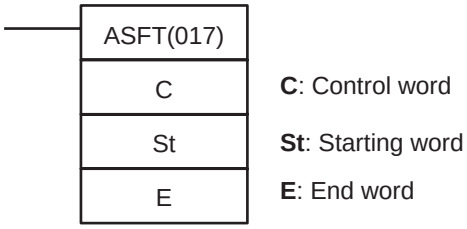


3-8-3 ASYNCHRONOUS SHIFT REGISTER: ASFT(017)

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

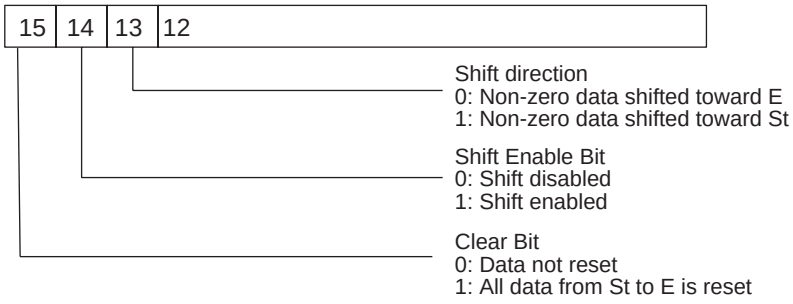
Variations		
	Executed Each Cycle for ON Condition	ASFT(017)
	Executed Once for Upward Differentiation	@ASFT(017)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

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.

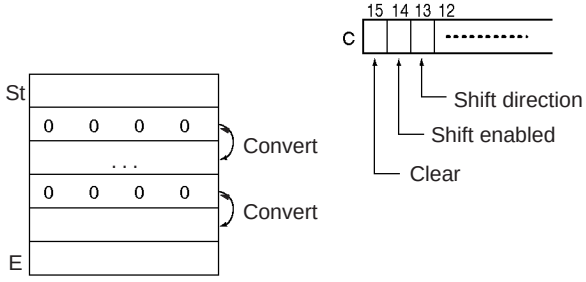
Operand Specifications

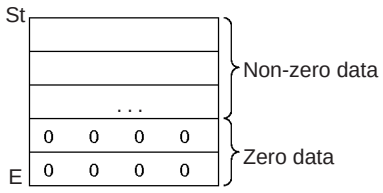
Area	C	St	E
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	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 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. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Precautions

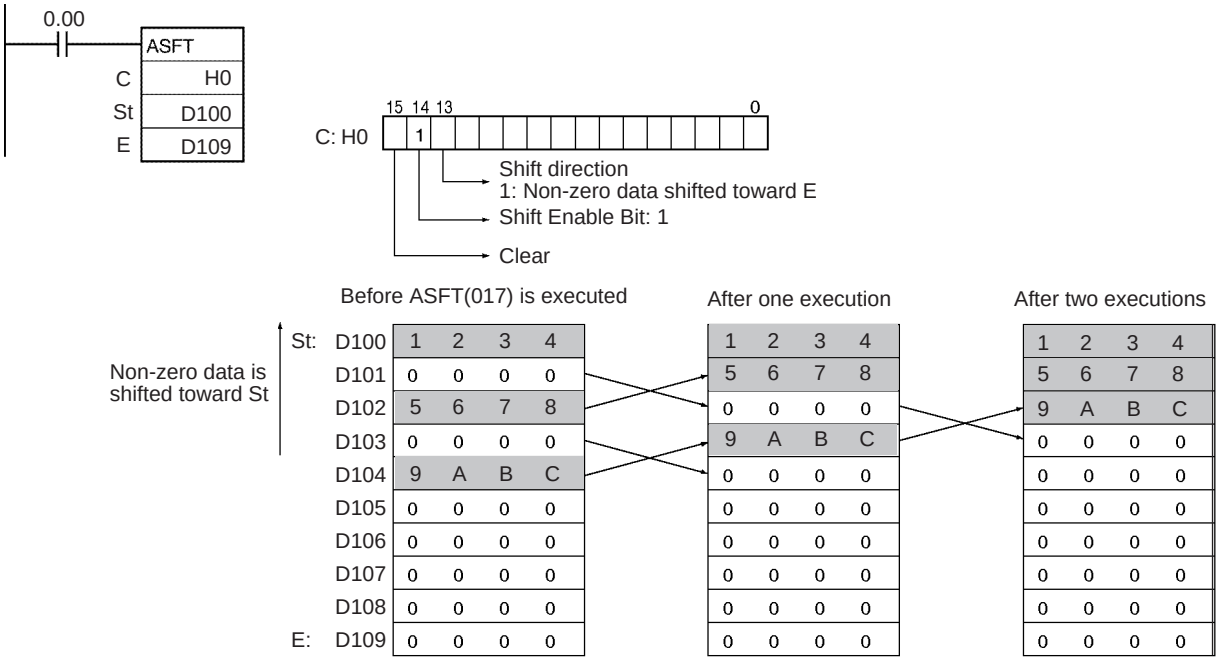
When the Clear Flag (bit 15 of C) goes ON, all bits in the shift register, from St to E, will be reset (i.e., set to 0). The Clear Flag has priority over the Shift Enable Bit (bit 14 of C).

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, H0.14, goes ON when CIO 0.00 is ON, all words with non-zero data content from D100 through D109 will be shifted in the direction designated by the Shift Direction Bit, H0.13 (e.g., 1: Toward St) if the word to the left of the non-zero data is all zeros.

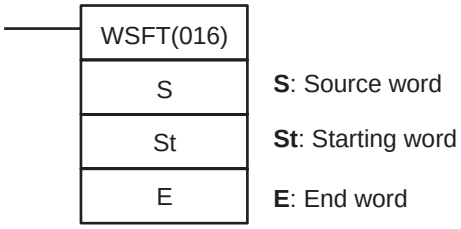


3-8-4 WORD SHIFT: WSFT(016)

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
Immediate Refreshing Specification		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.

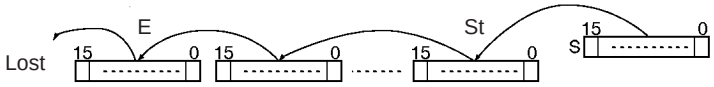
Operand Specifications

Area	S	St	E
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		

Area	S	St	E
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
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 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 places 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.

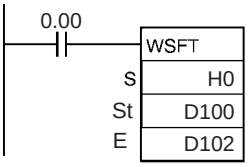
Precautions

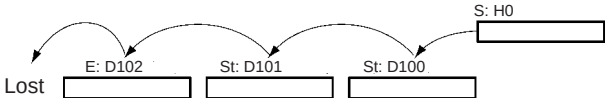
When St is greater than E, an error will be generated and the Error Flag will turn ON.

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 0.00 is ON, data from D100 through D102 will be shifted one word toward E. The contents of H0 will be stored in D100 and the contents of D102 will be lost.



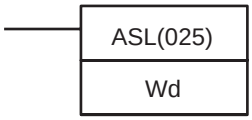


3-8-5 ARITHMETIC SHIFT LEFT: ASL(025)

Purpose

Shifts the contents of Wd one bit to the left.

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	ASL(025)
	Executed Once for Upward Differentiation	@ASL(025)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

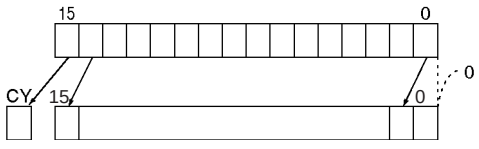
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

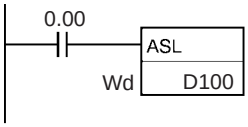
When ASL(025) is executed, the Error Flag will turn OFF.

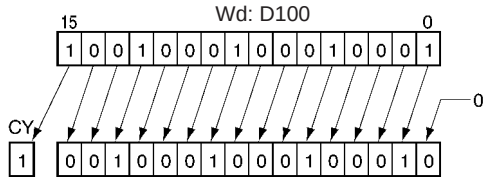
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Examples

When CIO 0.00 is ON, D100 will be shifted one bit to the left. “0” will be placed in bit 00 of D100 and the contents of bit 15 of D100 will be shifted to the Carry Flag (CY).



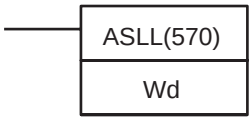


3-8-6 DOUBLE SHIFT LEFT: ASLL(570)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

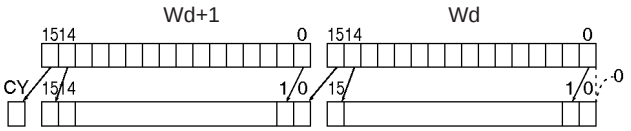
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510

Area	Wd
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 of Wd and Wd +1 are shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
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

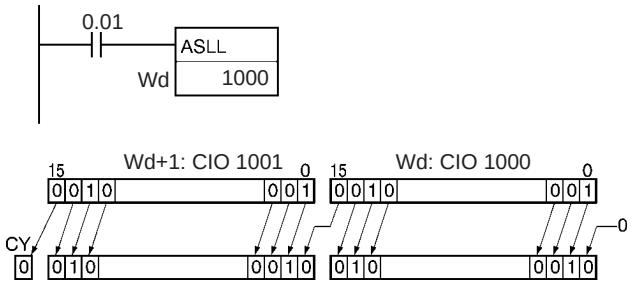
When ASLL(570) is executed, the Error Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0.01 is ON, word CIO 1000 and CIO 1001 will shift one bit to the left. “0” is placed into CIO 1000.00 and the contents of CIO 1001.15 will be shifted to the Carry Flag (CY).

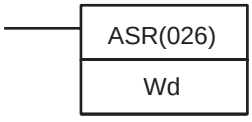


3-8-7 ARITHMETIC SHIFT RIGHT: ASR(026)

Purpose

Shifts the contents of Wd one bit to the right.

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	ASR(026)
	Executed Once for Upward Differentiation	@ASR(026)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

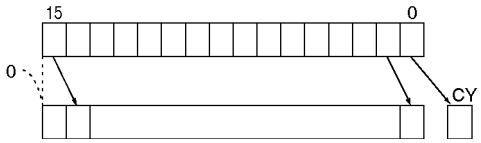
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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	OFF

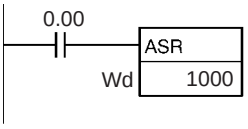
Precautions

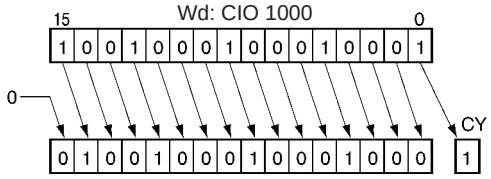
When ASR(026) is executed, the Error Flag and the Negative Flag will turn OFF.

If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

Examples

When CIO 0.00 is ON, word CIO 1000 will shift one bit to the right. “0” will be placed in CIO 100.15 and the contents of CIO 1000.00 will be shifted to the Carry Flag (CY).



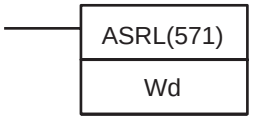


3-8-8 DOUBLE SHIFT RIGHT: ASRL(571)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

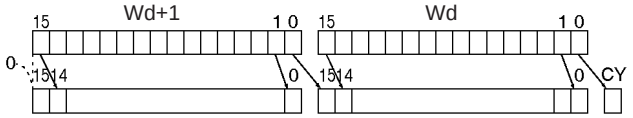
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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	OFF

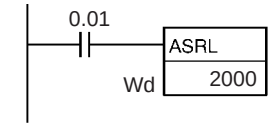
Precautions

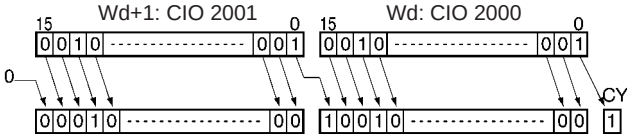
When ASRL (571) is executed, the Error Flag and the Negative Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

Examples

When CIO 0.01 is ON, word CIO 2000 and CIO 2001 will shift one bit to the right. “0” will be placed into CIO 2001.15 and the contents of CIO 2000.00 will be shifted to the Carry Flag (CY).



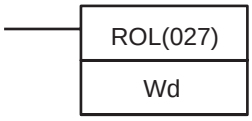


3-8-9 ROTATE LEFT: ROL(027)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

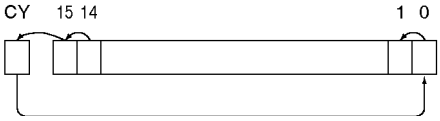
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

When ROL(027) is executed, the Error Flag will turn OFF.

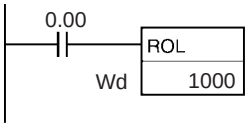
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

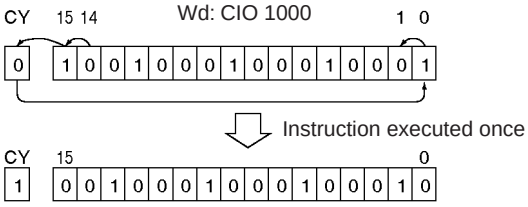
If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

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 0.00 is ON, word CIO 1000 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 1000.15 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 1000.00.



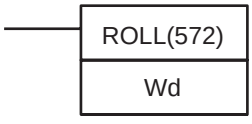


3-8-10 DOUBLE ROTATE LEFT: ROLL(572)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

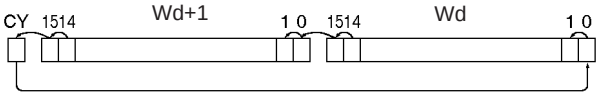
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

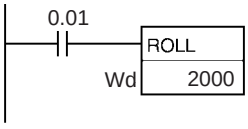
When ROLL(572) is executed, the Error Flag will turn OFF.

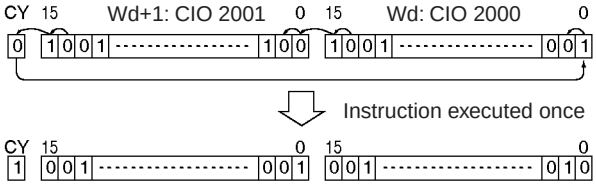
If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, 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 0.01 is ON, word CIO 2000, CIO 2001 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 2001.15 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 2000.00.

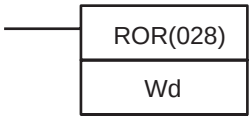




3-8-11 ROTATE RIGHT: ROR(028)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

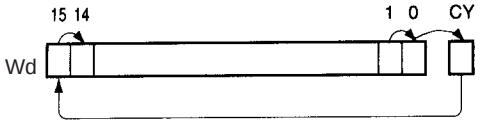
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---

Area	Wd
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

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 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

When ROR(028) is executed, the Error Flag will turn OFF.

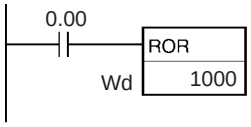
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

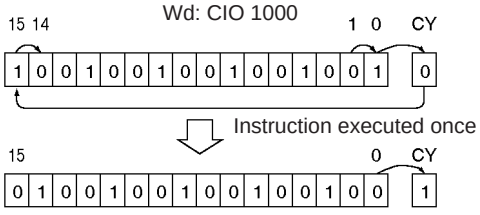
If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

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 0.00 is ON, word CIO 1000 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 1000.00 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 1000.15.



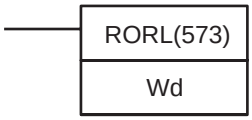


3-8-12 DOUBLE ROTATE RIGHT: RORL(573)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

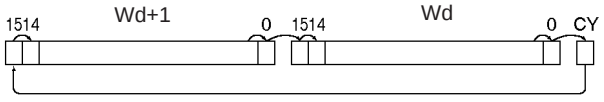
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

When RORL(573) is executed, the Error Flag will turn OFF.

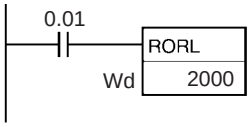
If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

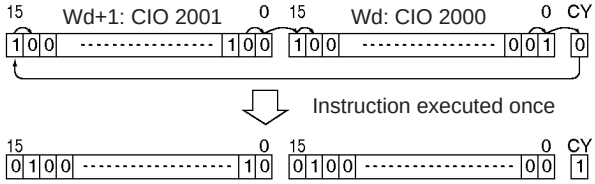
If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

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 0.01 is ON, word CIO 2000, CIO 2001 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 2000.00 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 2001.15.



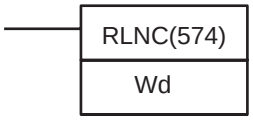


3-8-13 ROTATE LEFT WITHOUT CARRY: RLNC(574)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

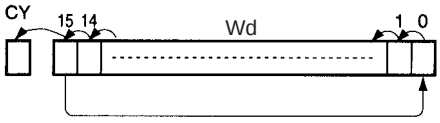
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511

Area	Wd
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

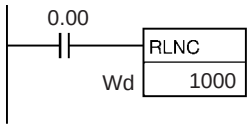
When RLNC(574) is executed, the Error Flag will turn OFF.

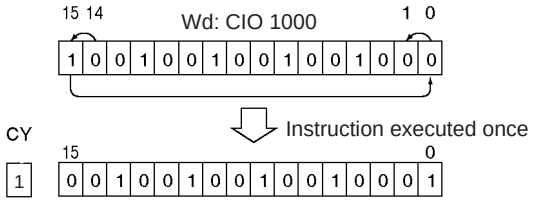
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Examples

When CIO 0.00 is ON, word CIO 1000 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 1000.15 will be shifted to CIO 1000.00.



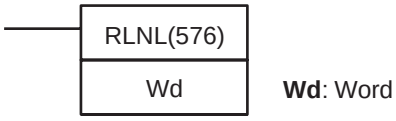


3-8-14 DOUBLE ROTATE LEFT WITHOUT CARRY: RLNL(576)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

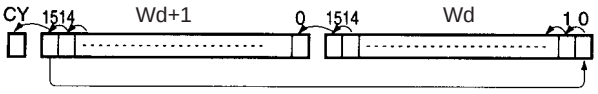
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	---

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 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 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

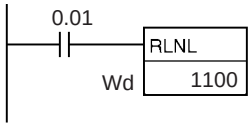
When RLNL(576) is executed, the Error Flag will turn OFF.

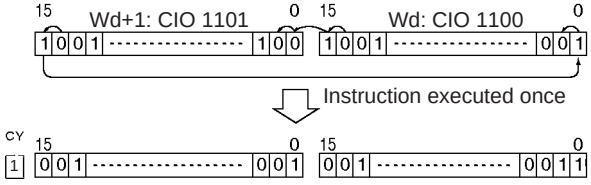
If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0.01 is ON, word CIO 1100 and CIO 1101 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 1101.15 will be shifted to CIO 1100.00.



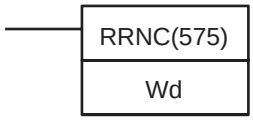


3-8-15 ROTATE RIGHT WITHOUT CARRY: RRNC(575)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

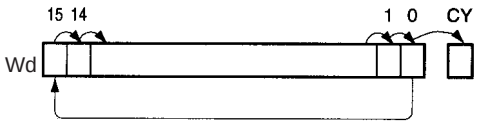
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

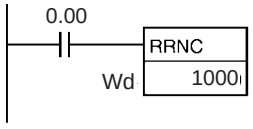
When RRNC(575) is executed, the Error Flag will turn OFF.

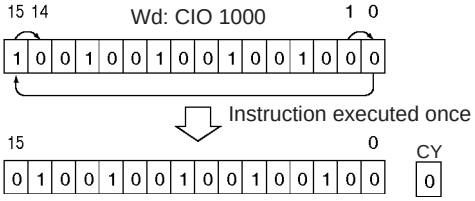
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Examples

When CIO 0.00 is ON, word CIO 1000 will shift one bit to the right (excluding the Carry Flag (CY)). The contents of CIO 1000.00 will be shifted to CIO 1000.15.



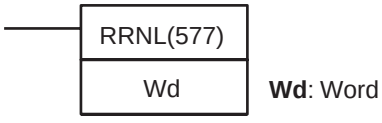


3-8-16 DOUBLE ROTATE RIGHT WITHOUT CARRY: RRNL(577)

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 +1 is shifted to the leftmost bit of Wd, and to the Carry Flag (CY).

Ladder Symbol



Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	RRNL(577)
	Executed Once for Upward Differentiation	@RRNL(577)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

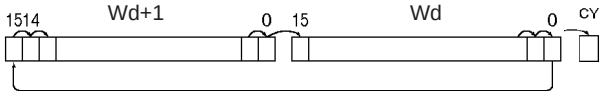
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 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 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

When RRNL(577) is executed, the Error Flag will turn OFF.

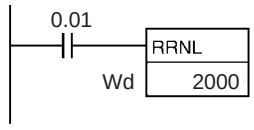
If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

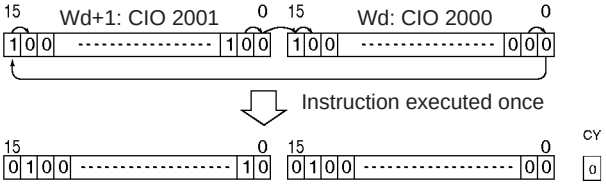
If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

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 0.01 is ON, words CIO 2000 and CIO 2001 will shift one bit to the right, (excluding the Carry Flag (CY)). The contents of CIO 2001.00 will be shifted to CIO 2000.15.



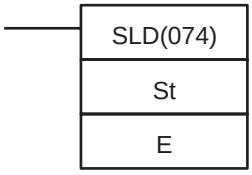


3-8-17 ONE DIGIT SHIFT LEFT: SLD(074)

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
Immediate Refreshing Specification		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.

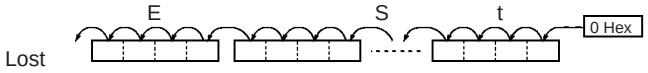
Operand Specifications

Area	St	E
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	
Data Registers	---	

Area	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 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.

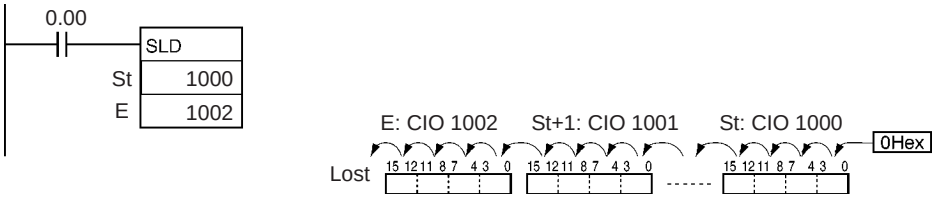
Precautions

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 0.00 is ON, words CIO 1000 through CIO 1002 will shift by one digit (4 bits) to the left. A zero will be placed in bits 0 to 3 of word CIO 1000 and the contents of bits 12 to 15 of CIO 1002 will be lost.

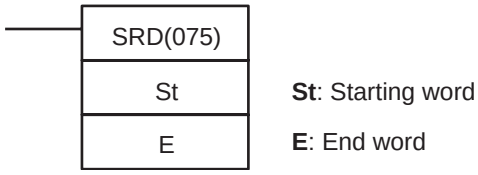


3-8-18 ONE DIGIT SHIFT RIGHT: SRD(075)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

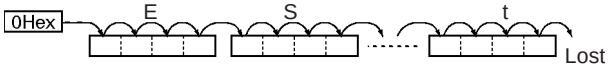
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

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.

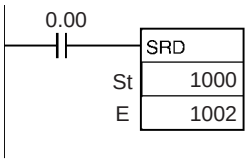
Precautions When St is greater than E, an error will be generated and the Error Flag will turn ON.

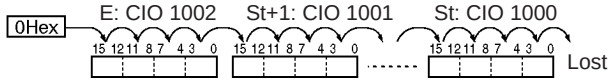
When SRD(075) is executed, the Equals Flag and Negative Flag will turn OFF.

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 0.00 is ON, words CIO 1000 through CIO 1002 will shift by one digit (4 bits) to the right. A zero will be placed in bits 12 to 15 of CIO 1002 and the contents of bits 0 to 3 of word CIO 1000 will be lost.



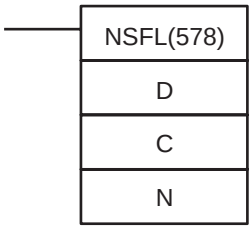


3-8-19 SHIFT N-BIT DATA LEFT: NSFL(578)

Purpose

Shifts the specified number of bits to the left.

Ladder Symbol



D: Beginning word for shift

C: Beginning bit

N: Shift data length

Variations

Variations		
	Executed Each Cycle for ON Condition	NSFL(578)
	Executed Once for Upward Differentiation	@NSFL(578)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: 0000 to 000F hex (0 to 15)
N: 0000 to FFFF hex (0 to 65535)

Note All words in the shift register must be in the same area.

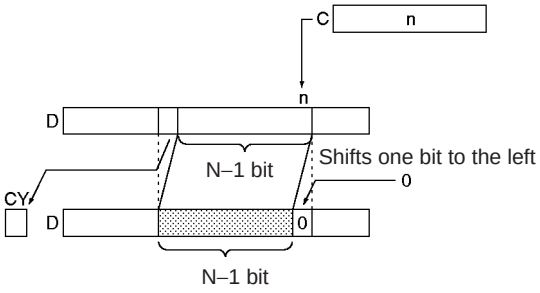
Operand Specifications

Area	D	C	N
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0000 to #000F (binary) or &0 to &15	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	---	DR0 to DR15	

Area	D	C	N
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

NSFL(578) shifts the specified number of bits by the shift data length (N) from the beginning bit (C) in the rightmost word, as designated by D one bit to the left (towards the leftmost word and the leftmost bit). “0” is place into the beginning bit and the contents of the leftmost bit in the shift area are shifted to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	ON when C data is not between 0000 and 000F hex. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.

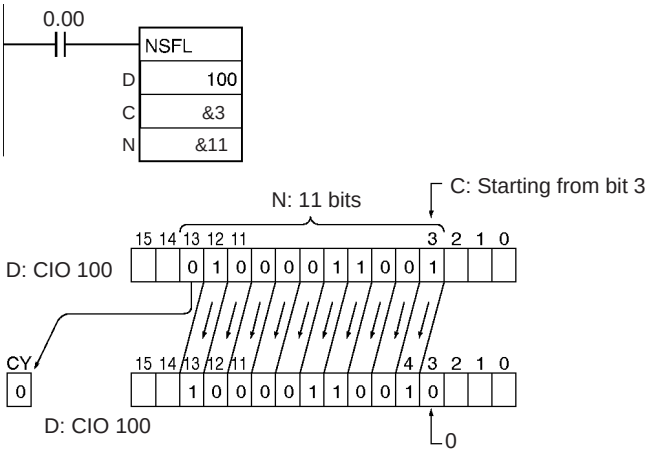
Precautions

When the shift data length (N) is 0, the contents of the beginning bit will be copied to the Carry Flag (CY), and its contents will not be changed.

Only the bits shifted into rightmost word in the shift area (i.e. leftmost word data) will be changed.

Examples

When CIO 0.00 is ON, all bits from the beginning bit 3 to the shift data length (B hex) will be shifted one bit to the left (from the rightmost bit to the leftmost bit). “0” will be placed into bit 3 of CIO 100. The contents of the leftmost bit in the shift area (bit 13 of CIO 100) are copied into the Carry Flag (CY).

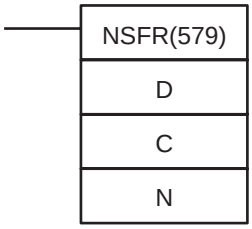


3-8-20 SHIFT N-BIT DATA RIGHT: NSFR(579)

Purpose

Shifts the specified number of bits to the right.

Ladder Symbol



D: Beginning word for shift

C: Beginning bit

N: Shift data length

Variations

Variations	**Executed Each Cycle for ON Condition**	NSFR(579)
	Executed Once for Upward Differentiation	@NSFR(579)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: 0000 to 000F hex (0 to 15)
N: 0000 to FFFF hex (0 to 65535)

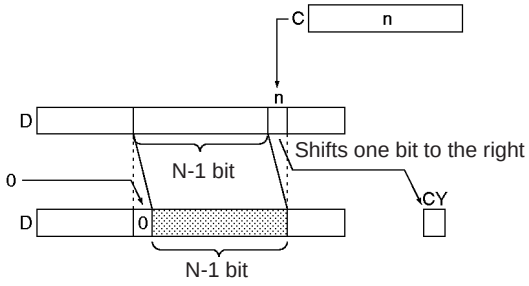
Note All words in the shift register must be in the same area.

Operand Specifications

Area	D	C	N
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

NSFR(579) shifts the specified number of bits by the shift data length (N) from the beginning bit (C) in the rightmost word as designated by D one bit to the right (towards the rightmost word and the rightmost bit). “0” will be placed into the beginning bit and the contents of the rightmost bit in the shift area will be shifted to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	ON when C data is not between 0000 and 000F hex. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.

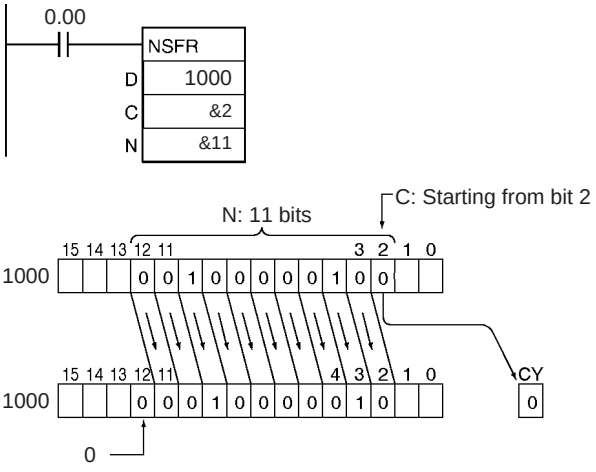
Precautions

When the shift data length (N) is 0, the contents of the beginning bit will be copied to the Carry Flag (CY), and its contents will not be changed.

Only the bits shifted into rightmost word in the shift area (i.e. leftmost word data) will be changed.

Examples

When CIO 0.00 is ON, all bits from the beginning bit 2 to end of the shift data length 11 bits (B hex), will be shifted one bit to the right , (from the leftmost bit to the rightmost bit). “0” is shifted into bit 12 of CIO 1000. The contents of the rightmost bit in the shift area (bit 2 of CIO 1000) are copied into the Carry Flag (CY).

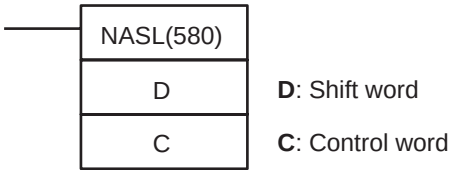


3-8-21 SHIFT N-BITS LEFT: NASL(580)

Purpose

Shifts the specified 16 bits of word data to the left by the specified number of bits.

Ladder Symbol



Variations

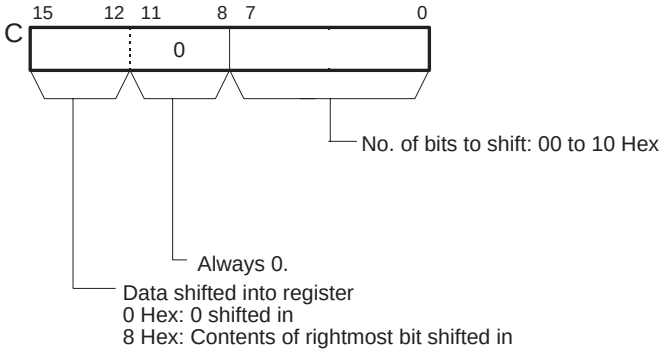
Variations	**Executed Each Cycle for ON Condition**	NASL(580)
	Executed Once for Upward Differentiation	@NASL(580)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word



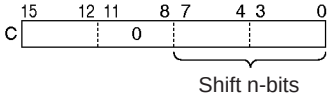
Operand Specifications

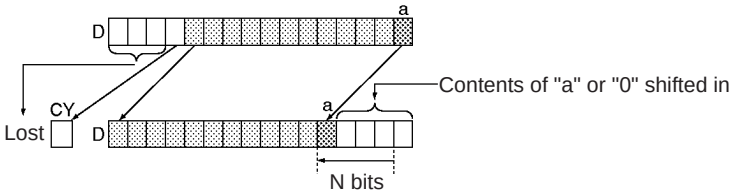
Area	D	C
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	A0 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	Specified values only
Data Registers	DR0 to DR15	

Area	D	C
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

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.

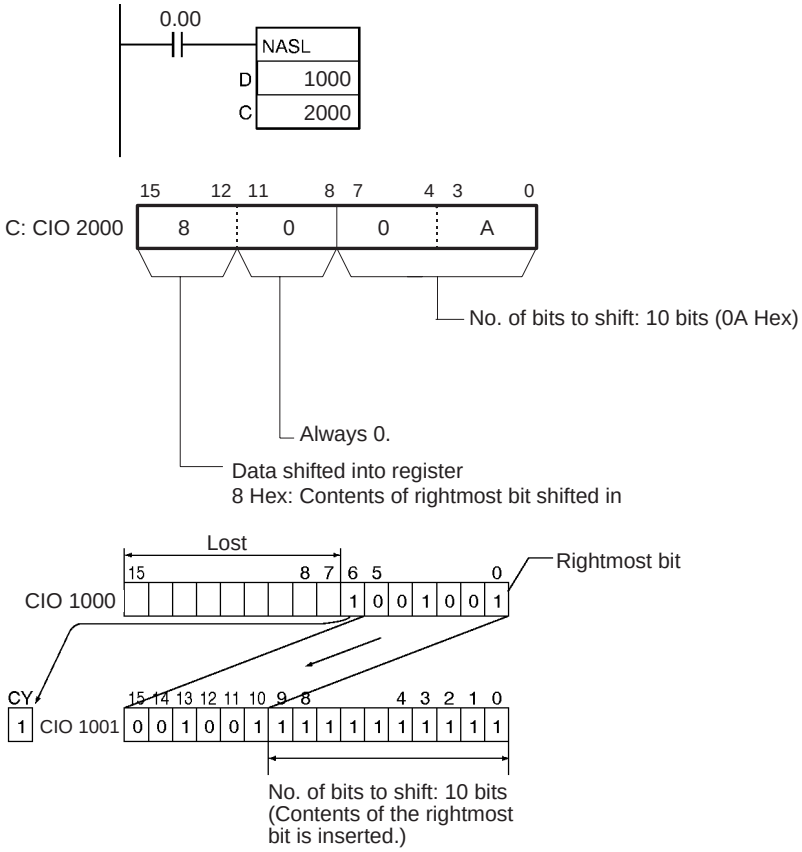
When the contents of the control word C is out of range, an error will be generated and the Error Flag will turn ON.

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 0.00 is ON, The contents of CIO 1000 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 2000 (control data). The contents of bit 0 of CIO 1001 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.

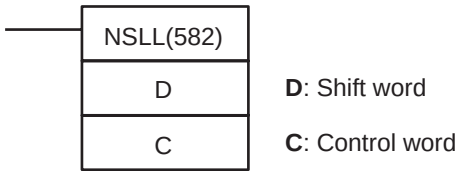


3-8-22 DOUBLE SHIFT N-BITS LEFT: NSLL(582)

Purpose

Shifts the specified 32 bits of word data to the left by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

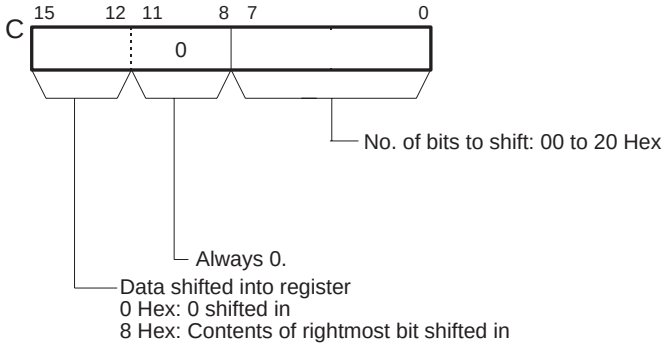
Variations	**Executed Each Cycle for ON Condition**	NSLL(582)
	Executed Once for Upward Differentiation	@NSLL(582)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

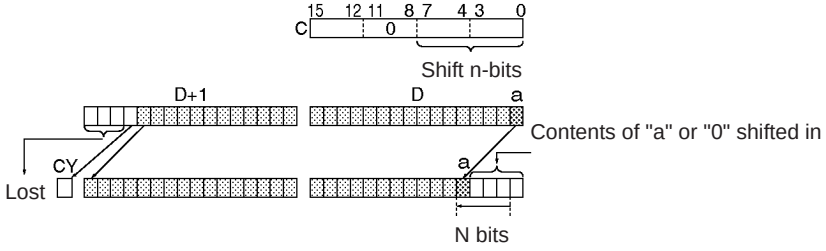


Operand Specifications

Area	D	C
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W510	W0 to W511
Holding Bit Area	H0 to H510	H0 to H511
Auxiliary Bit Area	A448 to A958	A0 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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

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.

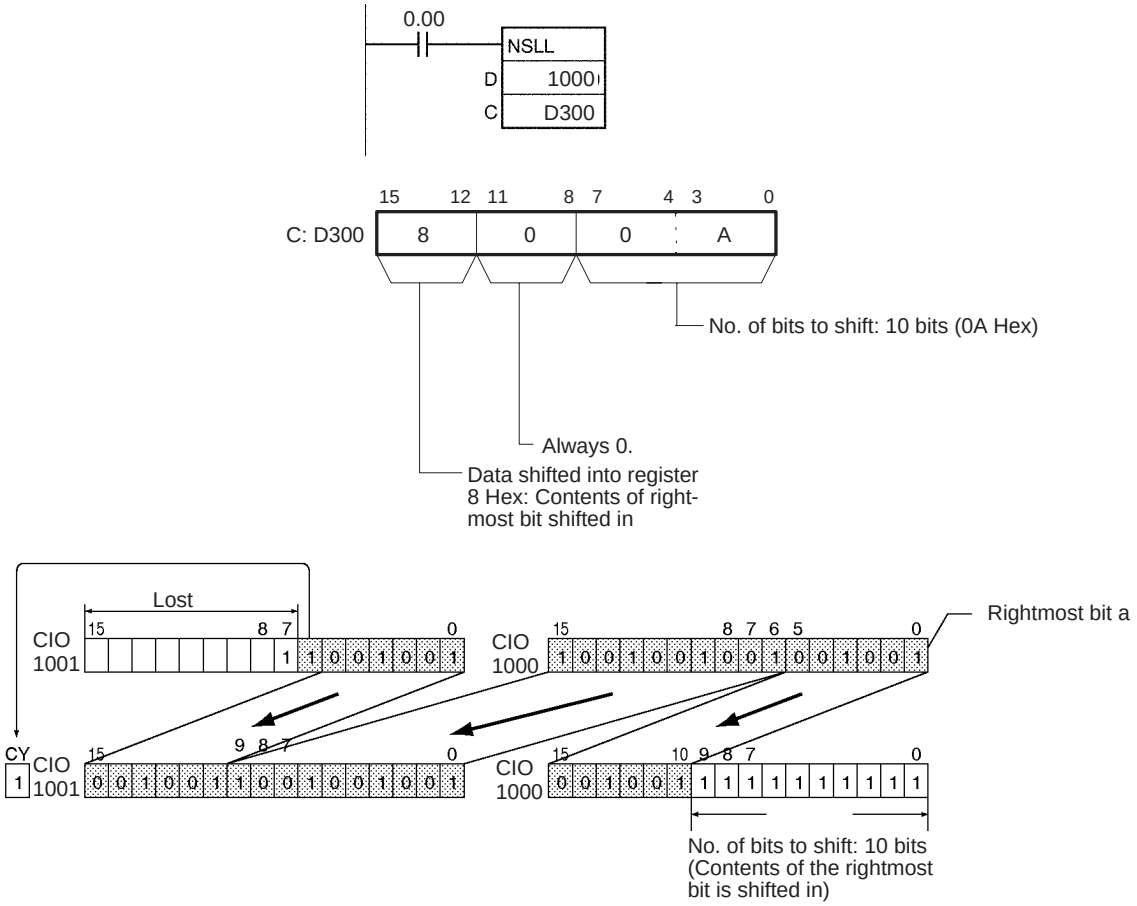
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

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 0.00 is ON, CIO 1000 and CIO 1001 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 D300 (control data). The contents of bit 0 of CIO 1000 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.

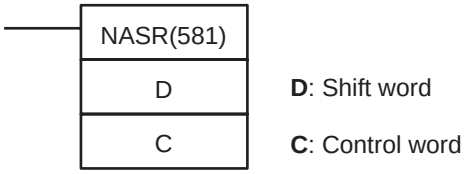


3-8-23 SHIFT N-BITS RIGHT: NASR(581)

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

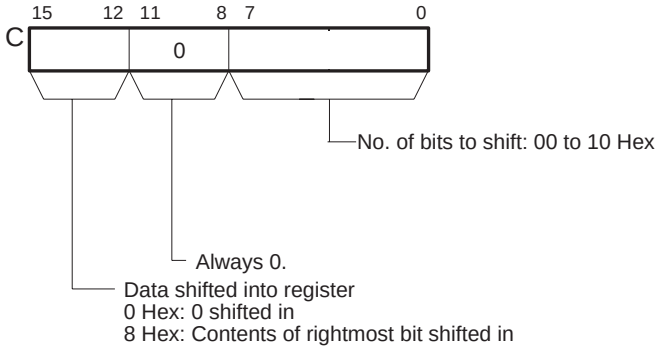
Variations		
	Executed Each Cycle for ON Condition	NASR(581)
	Executed Once for Upward Differentiation	@NASR(581)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

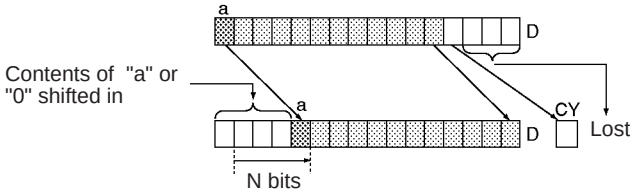


Operand Specifications

Area	D	C
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	A0 to A447 A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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

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.
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.

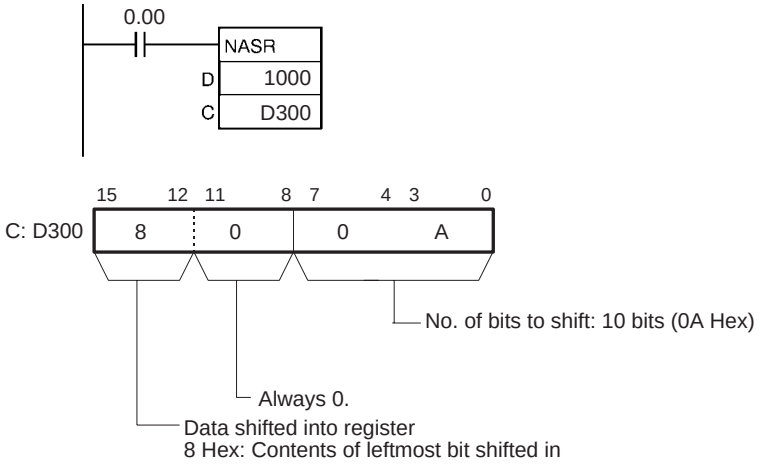
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

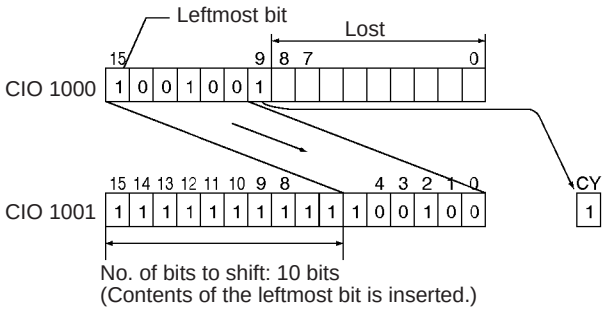
If as a result of the shift the contents of D is 0000 hex, the Equals Flag will turn ON.

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 0.00 is ON, CIO 1000 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 D300. The contents of bit 15 of CIO 1000 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.



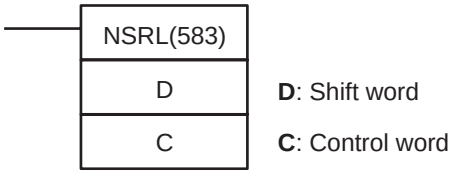


3-8-24 DOUBLE SHIFT N-BITS RIGHT: NSRL(583)

Purpose

Shifts the specified 32 bits of word data to the right by the specified number of bits.

Ladder Symbol



Variations

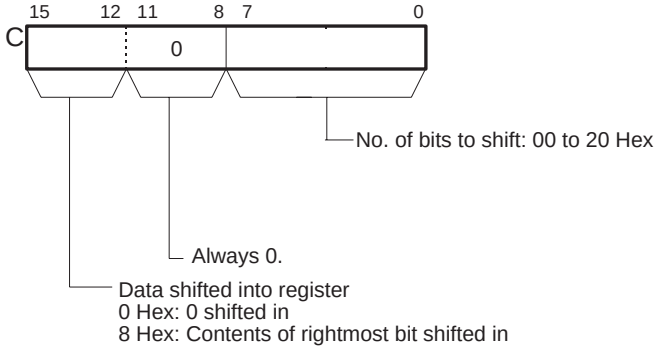
Variations	**Executed Each Cycle for ON Condition**	NSRL(583)
	Executed Once for Upward Differentiation	@NSRL(583)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word



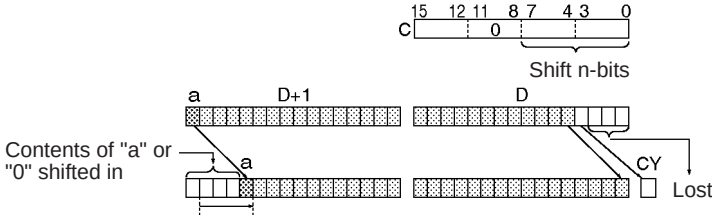
Operand Specifications

Area	D	C
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W510	W0 to W511
Holding Bit Area	H0 to H510	H0 to H511
Auxiliary Bit Area	A448 to A958	A0 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32766	D0 to D32767

Area	D	C
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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.

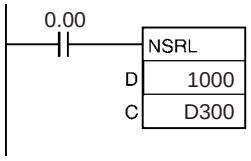
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

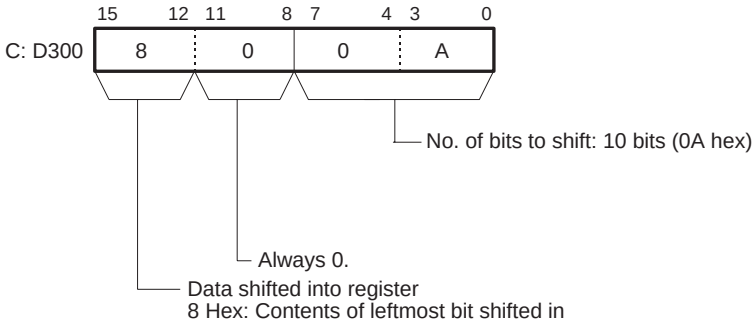
If as a result of the shift the contents of D +1 is 00000000 hex, the Equals Flag will turn ON.

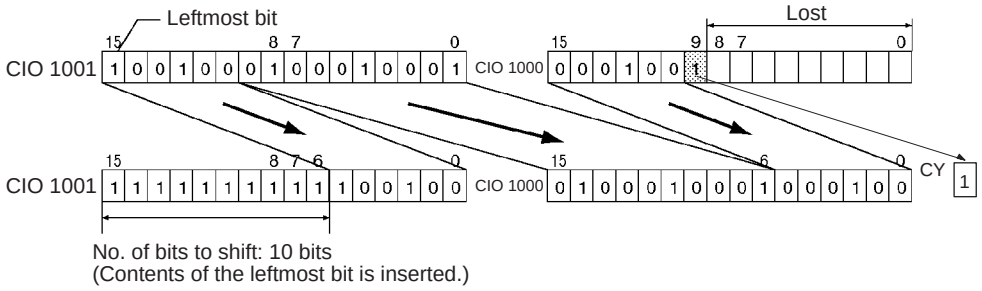
If as a result of the shift the contents of the leftmost bit of D +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0.00 is ON, CIO 1000 and CIO 1001 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 D300 (control data). The contents of bit 15 of CIO 1001 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.







3-9 Increment/Decrement Instructions

This section describes instructions used to increment data.

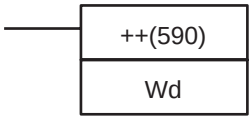
Instruction	Mnemonic	Function code	Page
INCREMENT BINARY	++	590	321
DOUBLE INCREMENT BINARY	++L	591	323
DECREMENT BINARY	– –	592	325
DOUBLE DECREMENT BINARY	– –L	593	327
INCREMENT BCD	++B	594	329
DOUBLE INCREMENT BCD	++BL	595	331
DECREMENT BCD	– –B	596	333
DOUBLE DECREMENT BCD	– –BL	597	335

3-9-1 INCREMENT BINARY: ++(590)

Purpose

Increments the 4-digit hexadecimal content of the specified word by 1.

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	++(590)
	Executed Once for Upward Differentiation	@++(590)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

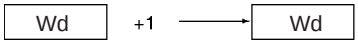
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	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 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, the Carry Flag will be turned ON when a digit changes from F to 0, 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.

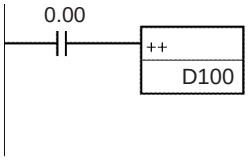
Flags

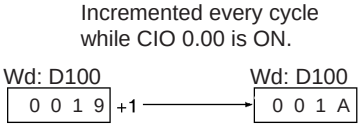
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd went from F to 0 during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd is ON after execution. OFF in all other cases.

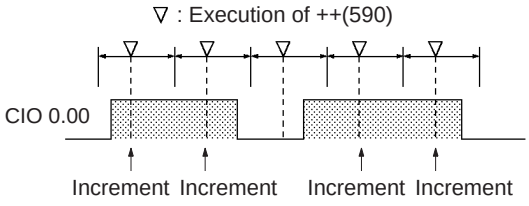
Examples

Operation of ++(590)

In the following example, the content of D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.

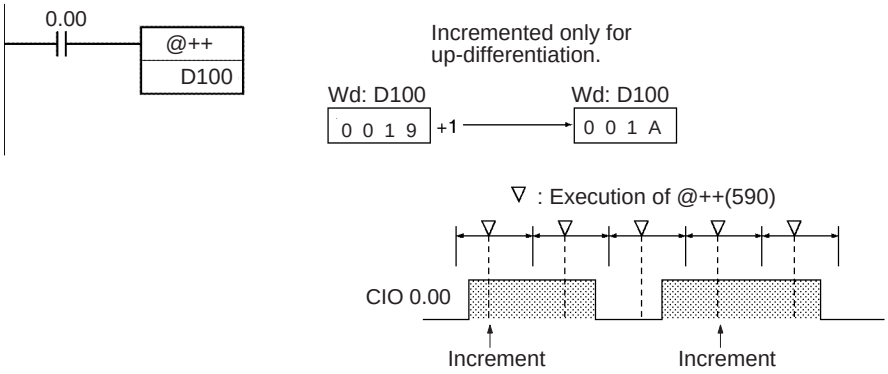






Operation of @++(590)

The up-differentiated variation is used in the following example, so the content of D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.

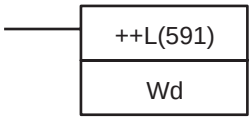


3-9-2 DOUBLE INCREMENT BINARY: ++L(591)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

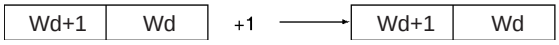
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	---

Area	Wd
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 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-differentiated 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, the Carry Flag will be turned ON when a digit changes from F to 0, 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 changes from FFFF FFFF to 0000 0000.

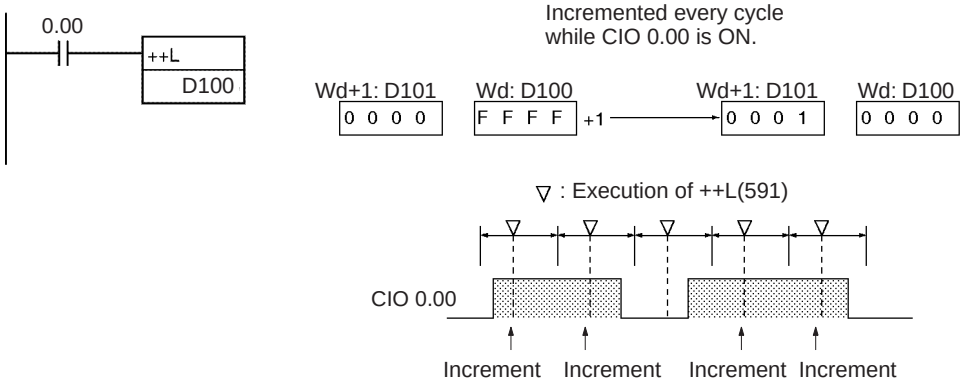
Flags

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 a digit in Wd+1 or Wd went from F to 0 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

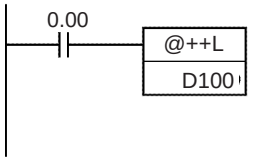
Operation of ++L(591)

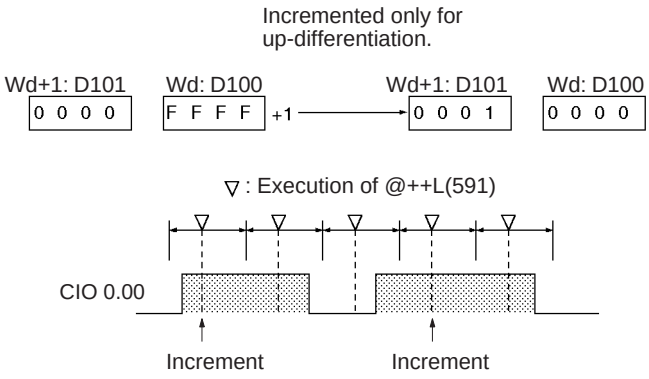
In the following example, the 8-digit hexadecimal content of D101 and D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.



Operation of @++L(591)

The up-differentiated variation is used in the following example, so the content of D101 and D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.



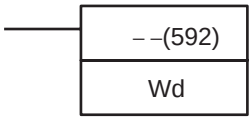


3-9-3 DECREMENT BINARY: – –(592)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

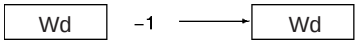
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	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 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, the Carry Flag will be turned ON when a digit changes from 0 to F, 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.

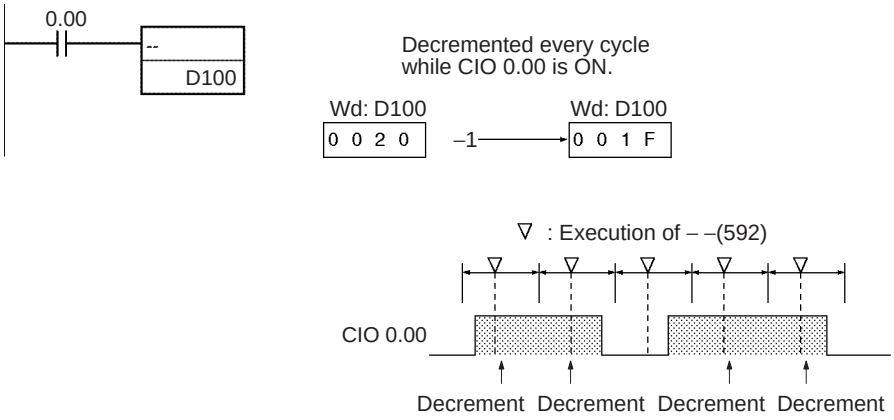
Flags

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 a digit in Wd went from 0 to F 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

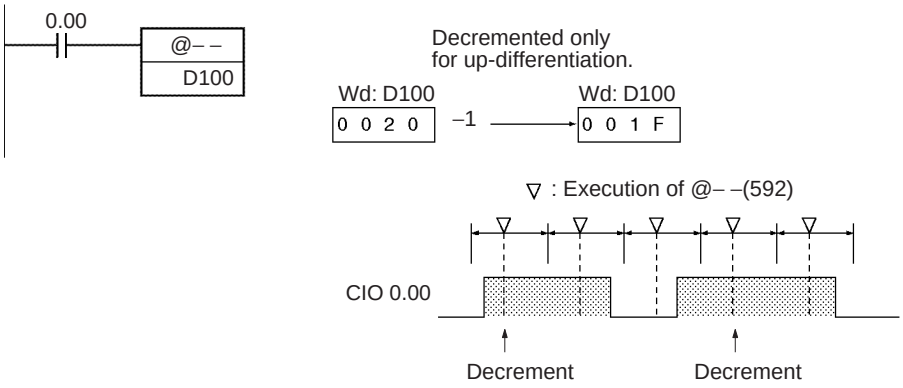
Operation of – –(592)

In the following example, the content of D100 will be decremented by 1 every cycle as long as CIO 0.00 is ON.



Operation of @– –(592)

The up-differentiated variation is used in the following example, so the content of D100 will be decremented by 1 only when CIO 0.00 has gone from OFF to ON.

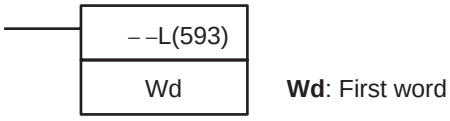


3-9-4 DOUBLE DECREMENT BINARY: – –L(593)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

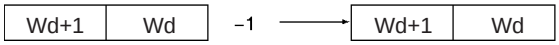
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	---

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 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, the Carry Flag will be turned ON when a digit changes from 0 to F, 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.

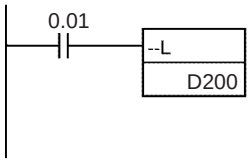
Flags

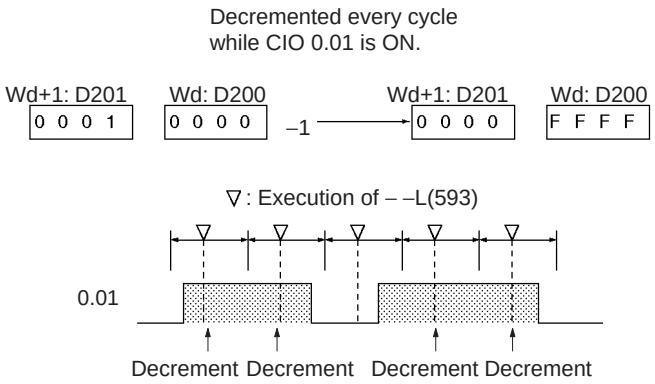
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd+1 or Wd went from 0 to F during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd+1 is ON after execution. OFF in all other cases.

Examples

Operation of – –L(593)

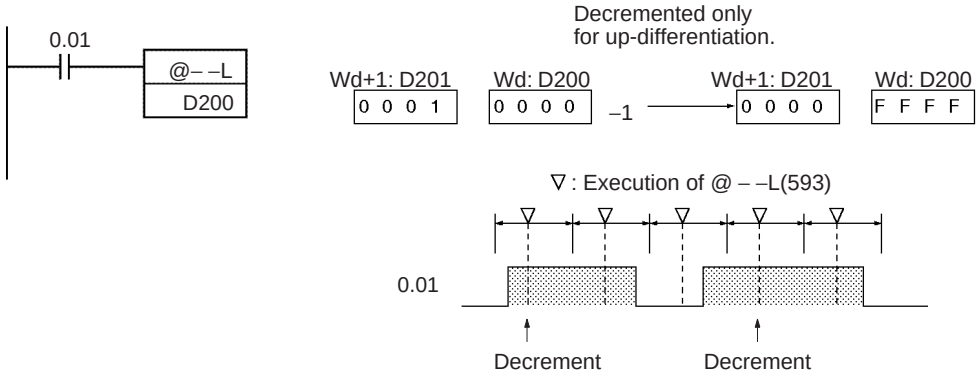
In the following example, the 8-digit hexadecimal content of D201 and D200 will be decremented by 1 every cycle as long as CIO 0.01 is ON.





Operation of @– –L(593)

The up-differentiated variation is used in the following example, so the content of D201 and D200 will be decremented by 1 only when CIO 0.01 has gone from OFF to ON.

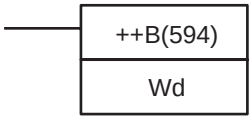


3-9-5 INCREMENT BCD: ++B(594)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

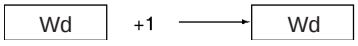
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in BCD	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	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 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 and the Carry Flag will be turned ON when a digit changes from 9 to 0.

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 a digit in Wd went from 9 to 0 during execution. OFF in all other cases.

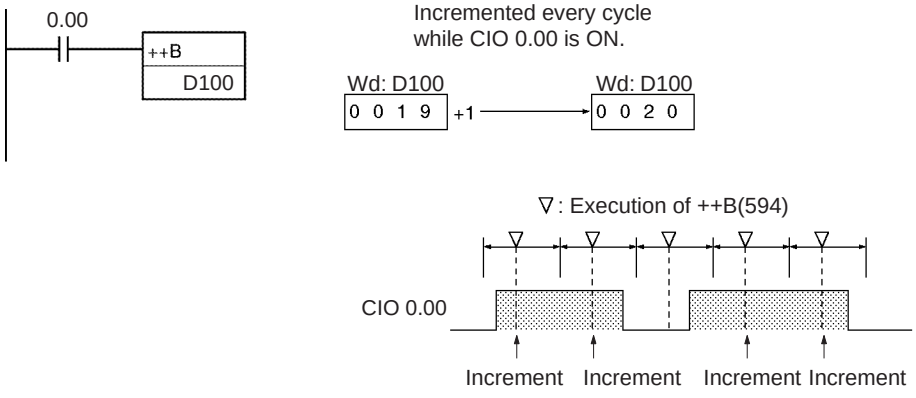
Precautions

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

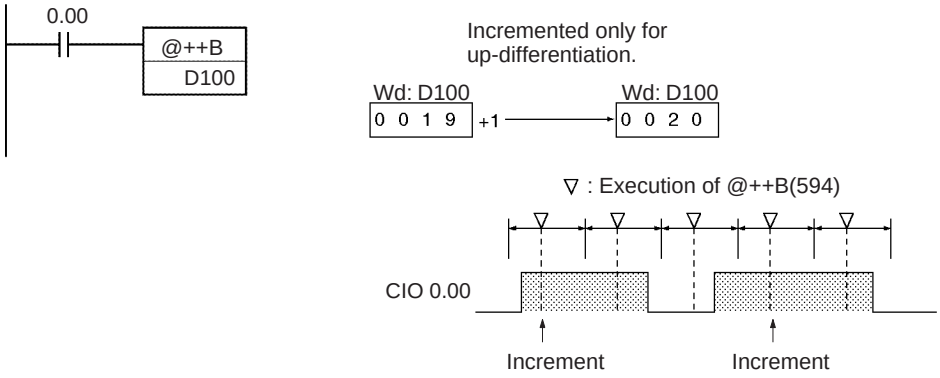
Operation of ++B(594)

In the following example, the BCD content of D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.



Operation of @++B(594)

The up-differentiated variation is used in the following example, so the content of D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.

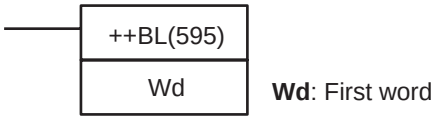


3-9-6 DOUBLE INCREMENT BCD: ++BL(595)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

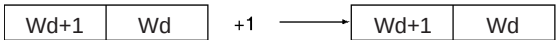
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in BCD	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	---

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 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 and the Carry Flag will be turned ON when a digit changes from 9 to 0.

Both the Equals Flag and the Carry Flag will be turned ON when the content of 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 a digit in Wd+1 or Wd went from 9 to 0 during execution. OFF in all other cases.

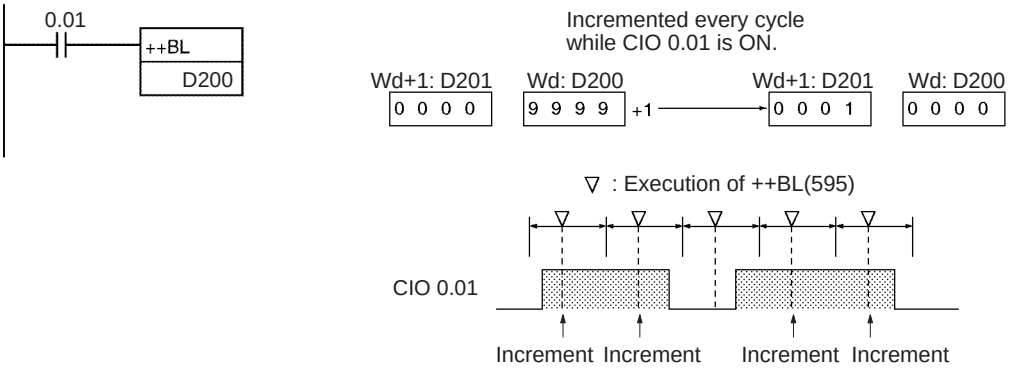
Precautions

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

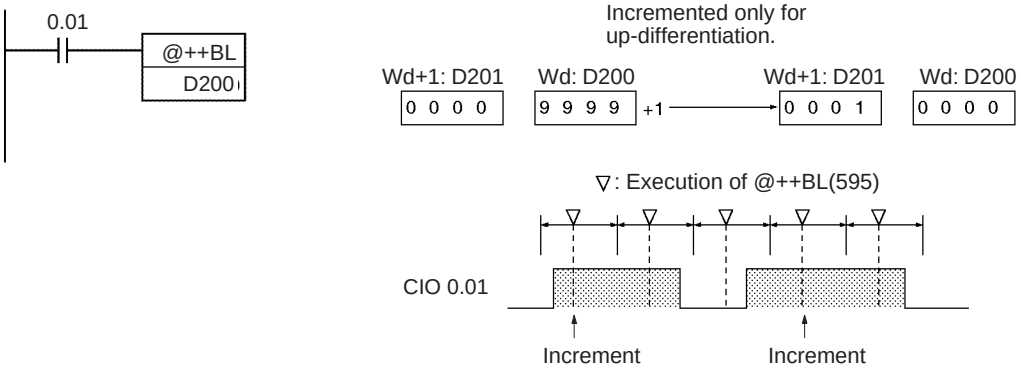
Operation of ++BL(595)

In the following example, the 8-digit BCD content of D201 and D200 will be incremented by 1 every cycle as long as CIO 0.01 is ON.



Operation of @++BL(595)

The up-differentiated variation is used in the following example, so the BCD content of D201 and D200 will be incremented by 1 only when CIO 0.01 has gone from OFF to ON.

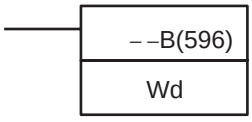


3-9-7 DECREMENT BCD: – –B(596)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

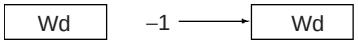
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in BCD	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	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 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 a digit changes from 0 to 9.

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 a digit in Wd went from 0 to 9 during execution. OFF in all other cases.

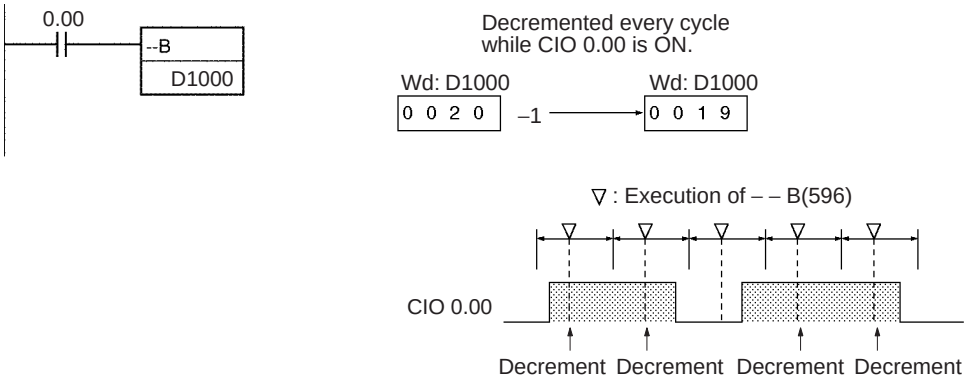
Precautions

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

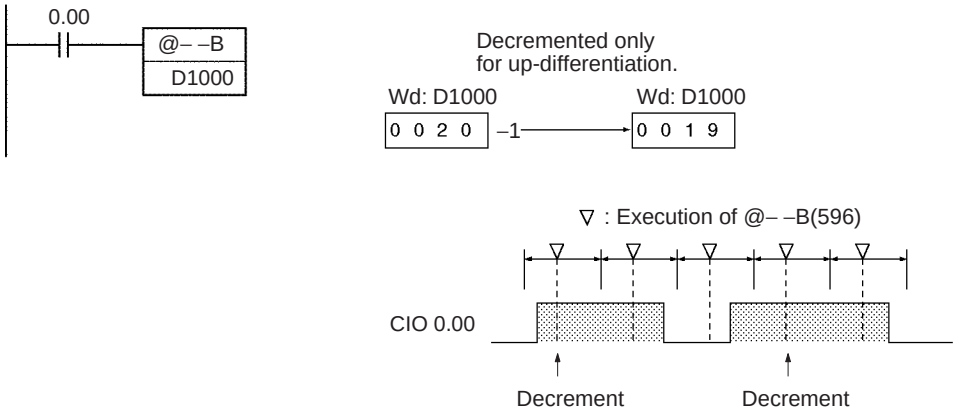
Operation of – –B(596)

In the following example, the BCD content of D1000 will be decremented by 1 every cycle as long as CIO 0.00 is ON.



Operation of @– –B(596)

The up-differentiated variation is used in the following example, so the BCD content of D1000 will be decremented by 1 only when CIO 0.00 has gone from OFF to ON.

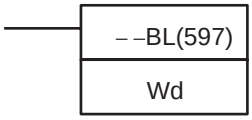


3-9-8 DOUBLE DECREMENT BCD: – –BL(597)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

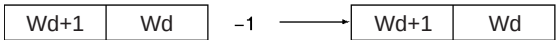
Operand Specifications

Area	Wd
CIO Area	CIO 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in BCD	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	---
Data Registers	---

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 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 a digit changes from 0 to 9.

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 a digit in Wd+1 or Wd went from 0 to 9 during execution. OFF in all other cases.

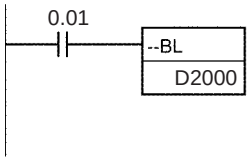
Precautions

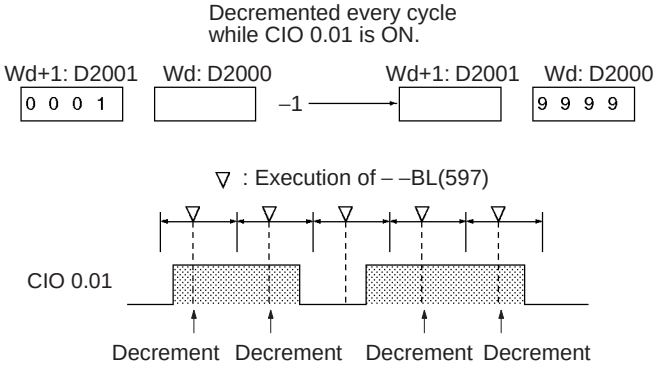
The content of Wd+1 and Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

Operation of – –BL(597)

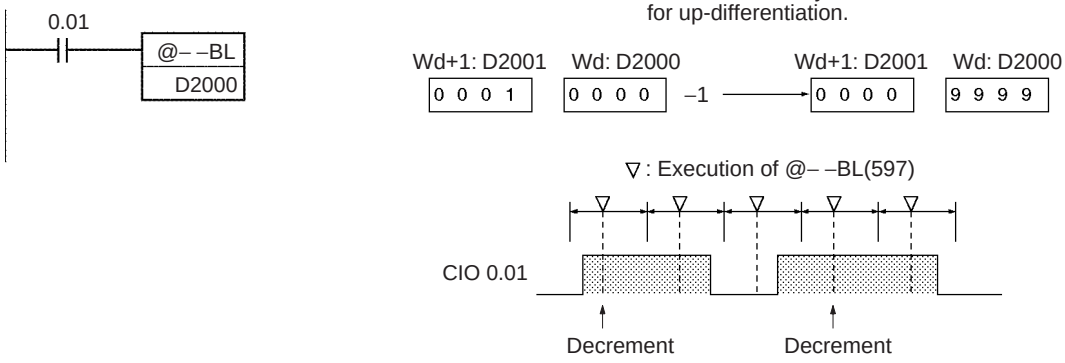
In the following example, the 8-digit BCD content of D2001 and D2000 will be decremented by 1 every cycle as long as CIO 0.01 is ON.





Operation of @– –BL(597)

The up-differentiated variation is used in the following example, so the BCD content of D2001 and D2000 will be decremented by 1 only when CIO 0.01 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	338
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	340
SIGNED BINARY ADD WITH CARRY	+C	402	342
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	344
BCD ADD WITHOUT CARRY	+B	404	346
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	347
BCD ADD WITH CARRY	+BC	406	349
DOUBLE BCD ADD WITH CARRY	+BCL	407	350
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	352
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	354
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	358
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	360
BCD SUBTRACT WITHOUT CARRY	–B	414	362
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	364
BCD SUBTRACT WITH CARRY	–BC	416	367
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	368
SIGNED BINARY MULTIPLY	*	420	370

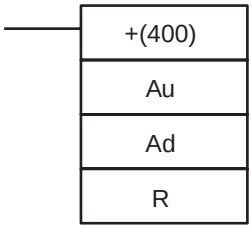
Instruction	Mnemonic	Function code	Page
DOUBLE SIGNED BINARY MULTIPLY	*L	421	372
UNSIGNED BINARY MULTIPLY	*U	422	373
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	375
BCD MULTIPLY	*B	424	376
DOUBLE BCD MULTIPLY	*BL	425	378
SIGNED BINARY DIVIDE	/	430	379
DOUBLE SIGNED BINARY DIVIDE	/L	431	381
UNSIGNED BINARY DIVIDE	/U	432	383
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	385
BCD DIVIDE	/B	434	386
DOUBLE BCD DIVIDE	/BL	435	388

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

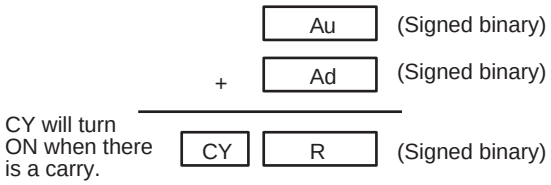
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		

Area	Au	Ad	R
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)		---
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

+(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 0. 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.

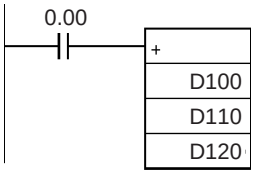
If the result of adding two positive numbers is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

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 0.00 is ON in the following example, D100 and D110 will be added as 4-digit signed binary values and the result will be output to D120.

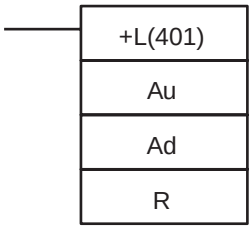


3-10-2 DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

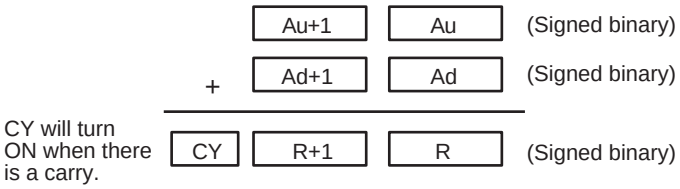
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal) –2147483648 to 2147483647 (signed decimal)		---
Data Registers	---		

Area	Au	Ad	R
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 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.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. 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 80000000 to FFFFFFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers is in the range 00000000 to 7FFFFFFF 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 00000000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

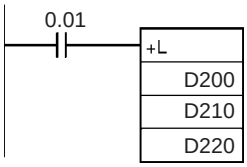
If the result of adding two positive numbers is negative (in the range 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

If the result of adding two negative numbers is positive (in the range 00000000 to 7FFFFFFF 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 0.01 is ON, D200 and D201 and D211 and D210 will be added as 8-digit signed binary values and the result will be output to D221 and D220.

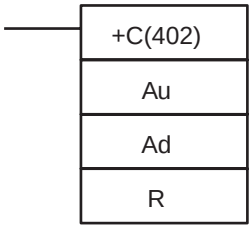


3-10-3 SIGNED BINARY ADD WITH CARRY: +C(402)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

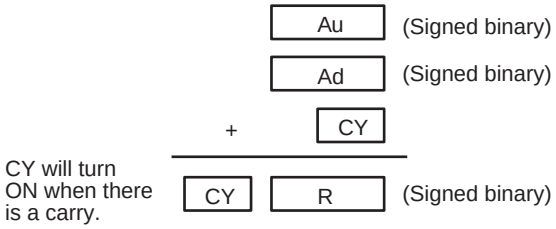
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)		---
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

+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 0. 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 addition 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 addition 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.

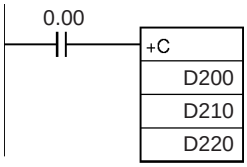
If the result of adding two negative numbers and CY is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

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 0.00 is ON, D200, D210, and CY will be added as 4-digit signed binary values and the result will be output to D220.

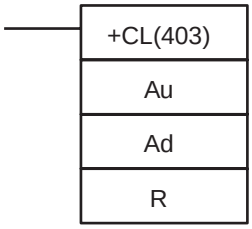


3-10-4 DOUBLE SIGNED BINARY ADD WITH CARRY: +CL(403)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

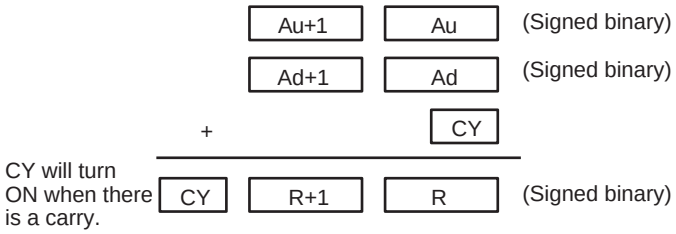
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal) –2147483648 to 2147483647 (signed decimal)		---
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 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.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the 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 80000000 to FFFFFFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers and CY is in the range 00000000 to 7FFFFFFF 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 00000000 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 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

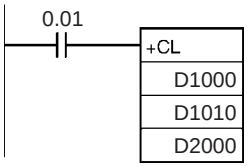
If the result of adding two negative numbers and CY is positive (in the range 00000000 to 7FFFFFFF hex), the Underflow Flag will turn ON.

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 0.01 is ON, D1001, D1000, D1011, D1010, and CY will be added as 8-digit signed binary values, and the result will be output to D2001 and D2000.

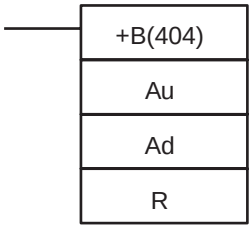


3-10-5 BCD ADD WITHOUT CARRY: +B(404)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

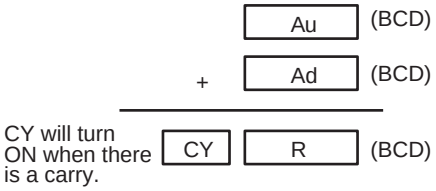
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	0000 to 9999 (BCD)		---
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

+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 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

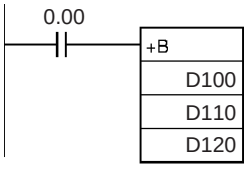
If Au or Ad is not BCD, an error is generated and the Error Flag will turn ON.

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 0.00 is ON in the following example, D100 and D110 will be added as 4-digit BCD values, and the result will be output to D120.

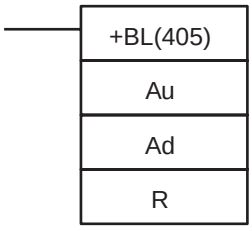


3-10-6 DOUBLE BCD ADD WITHOUT CARRY: +BL(405)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

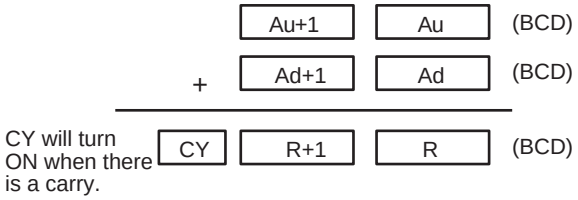
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		

Area	Au	Ad	R
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #99999999 (BCD)		---
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 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 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

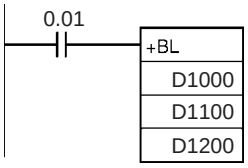
If Au, Au +1 or Ad, Ad +1 are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the addition, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, D1001 and D1000 and D1101 and D1100 will be added as 8-digit BCD values, and the result will be output to D1201 and D1200.

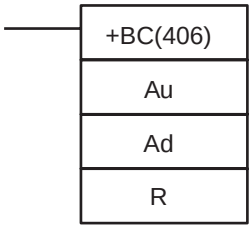


3-10-7 BCD ADD WITH CARRY: +BC(406)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

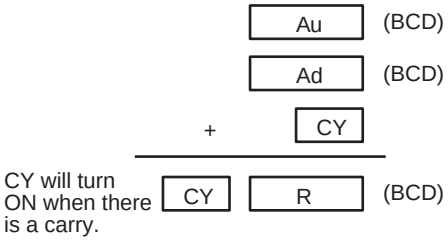
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to 9999 (BCD)		---
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

+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 0. 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.

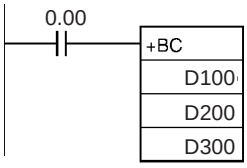
If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Note To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0.00 is ON in the following example, D100, D200, and CY will be added as 4-digit BCD values, and the result will be output to D300.

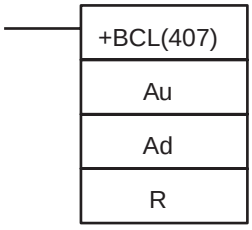


3-10-8 DOUBLE BCD ADD WITH CARRY: +BCL(407)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

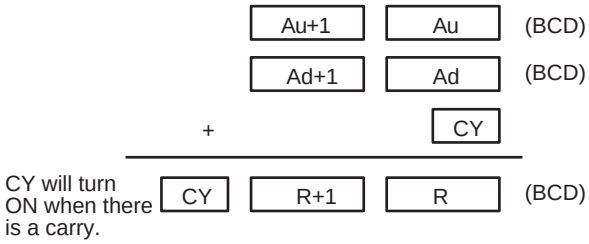
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958

Area	Au	Ad	R
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #99999999 (BCD)		---
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 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 0. 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.

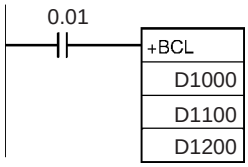
If as a result of the addition, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Note To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0.01 is ON in the following example, D1001, D1000, D1101, D1100, and CY will be added as 8-digit BCD values, and the result will be output to D1201 and D1200.

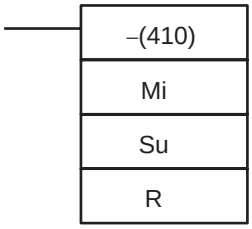


3-10-9 SIGNED BINARY SUBTRACT WITHOUT CARRY: –(410)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

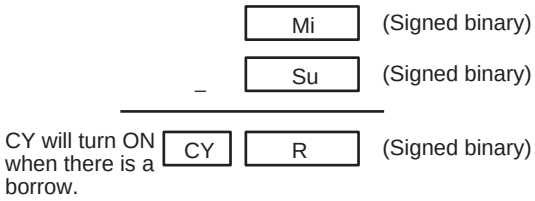
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D4095		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 32767 (signed decimal)		---
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

–(400) 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 0. 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 negative number from a positive 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.

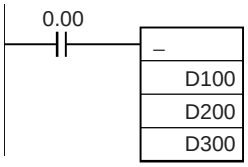
If the result of subtracting a negative number from a positive number is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

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 0.00 is ON in the following example, D200 will be subtracted from D100 as 4-digit signed binary values and the result will be output to D300.

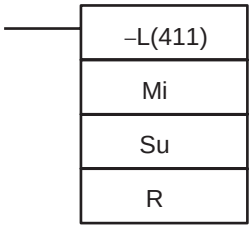


3-10-10 DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411)

Purpose

Subtracts 8-digit (double-word) hexadecimal data and/or constants.

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

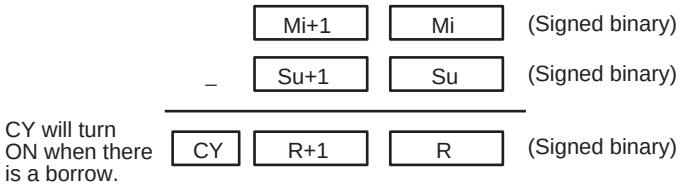
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal) –2147483648 to 2147483647 (signed decimal)		---
Data Registers	---		
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 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 0. 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 80000000 to FFFFFFFF 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 00000000 to 7FFFFFFF 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 00000000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

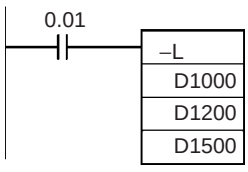
If the result of subtracting a negative number from a positive number is negative (in the range 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

If the result of subtracting a positive number from a negative number is positive (in the range 00000000 to 7FFFFFFF 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 0.01 is ON in the following example, D1201 and D1200 will be subtracted from D1001 and D1000 as 8-digit signed binary values and the result will be output to D1501 and D1500.



Examples

If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), 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 0 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 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 from the 2's complement b (hexadecimal): a (hexadecimal) = 10000 (hexadecimal) – b (hexadecimal). For example, to obtain the true number 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

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.

Example 2

	Signed data	Unsigned data
FFFD Hex →	–3	65533
–)FFFF Hex →	–) –1	–) 65535
FFFE Hex →	–2 Note 3	65534 Note 4

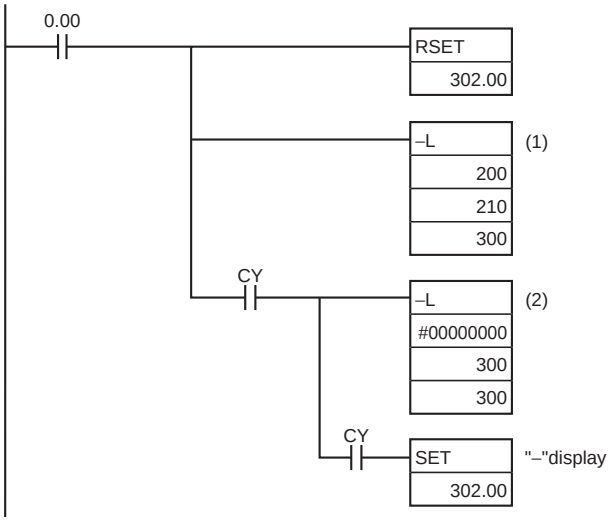
Negative Flag ON
Carry Flag OFF

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 211 and CIO 210 is subtracted from the value in CIO 201 and CIO 200, and the result is output in eight-digit binary to CIO 301 and CIO 300. If the result is negative, the instruction at (2) will be executed, and the actual result will then be output to CIO 301 and CIO 300.



Subtraction at 1

	Mi+1: CIO 201	Mi: CIO 200
	2 0 F 5	5 A 1 0
	Su+1: CIO 121	Su: CIO 120
–	B 8 A 3	6 0 E 3

CY	R+1: D101	R+1: D100
1	6 8 5 1	F 9 2 D

The Carry Flag (CY) is ON, so the result is subtracted from 0000 0000 to obtain the actual number.

Subtraction at 2

	0 0 0 0	0 0 0 0
	Su+1: CIO 201	Su: CIO 200
–	6 8 5 1	F 9 2 D

CY	R+1: CIO 201	R+1: CIO 200
1	9 7 A E	0 6 D 3

Final Subtraction Result

	Mi+1: CIO 201	Mi: CIO 200
	2 0 F 5	5 A 1 0
	Su+1: CIO 211	Su: CIO210
–	6 8 5 1	F 9 2 D

CY	R+1: CIO 301	R+1: CIO 300
1	9 7 A E	0 6 D 3

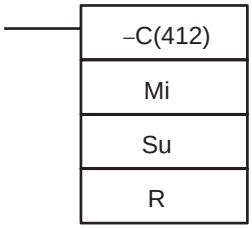
The Carry Flag (CY) is turned ON, so the actual number is –97AE06D3. Because the content of CIO 301 and CIO 300 is negative, CY is used to turn ON CIO 302.00 to indicate this.

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



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	–C(412)
	Executed Once for Upward Differentiation	@–C(412)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

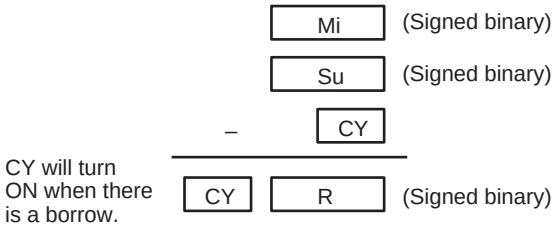
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)		---
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

–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.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the subtraction result is 0. 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.

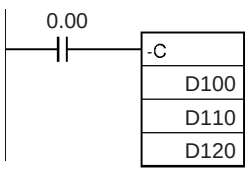
If the result of subtracting a positive number and CY from a negative number is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

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 0.00 is ON in the following example, D110 and CY will be subtracted from D100 as 4-digit signed binary values and the result will be output to D120.

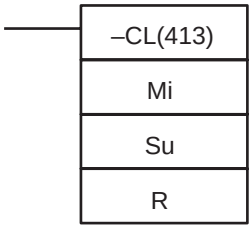


3-10-12 DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: –CL(413)

Purpose

Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

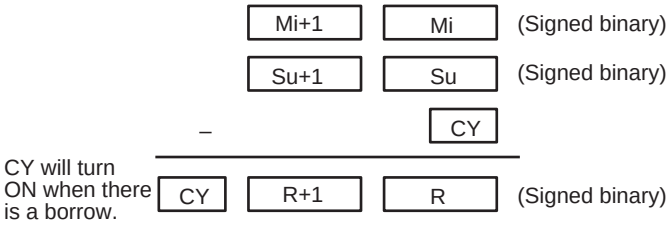
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal) –2147483648 to 2147483647 (signed decimal)		---
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 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 0. OFF in all other cases.
Carry Flag	CY	ON when the 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 80000000 to FFFFFFFF 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 00000000 to 7FFFFFFF 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 00000000 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 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

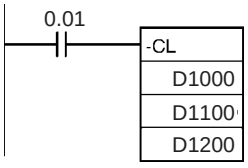
If the result of subtracting a positive number and CY from a negative number is positive (in the range 00000000 to 7FFFFFFF hex), the Underflow Flag will turn ON.

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 0.01 is ON in the following example, D1101, D1100 and CY will be subtracted from D1001 and D1000 as 8-digit signed binary values, and the result will be output to D1201 and D1200.



If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), 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 0 is necessary, as an input condition of the Carry Flag (CY). 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 value "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 CFC& hex:

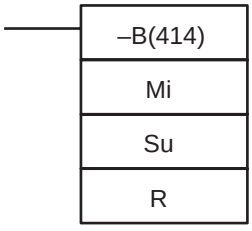
10000 hex – CFC7 hex = 3039 hex.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

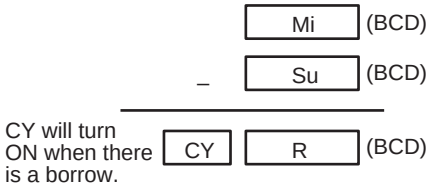
Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		

Area	Mi	Su	R
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	0000 to 9999 (BCD)		---
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

–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 as a 10's complement.



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 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

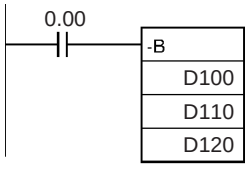
If Mi and/or Su are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a borrow, the Carry Flag will turn ON.

Examples

When CIO 0.00 is ON in the following example, D110 will be subtracted from D100 as 4-digit BCD values, and the result will be output to D120.

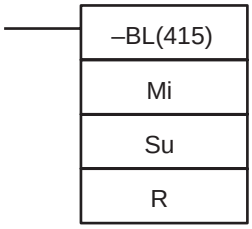


3-10-14 DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

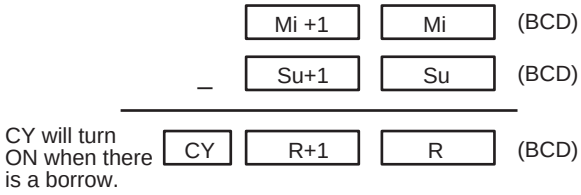
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #99999999 (BCD)		---
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 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 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

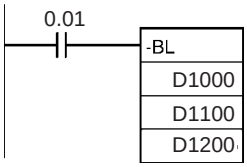
If Mi, Mi +1 and/or Su, Su +1 are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the subtraction, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an addition results in a borrow, the Carry Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, D1001 and D1000 will be subtracted from D1101 and D1100 as 8-digit BCD values, and the result will be output to D1201 and D1200.



If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), 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, as an input condition of the Carry Flag (CY). The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

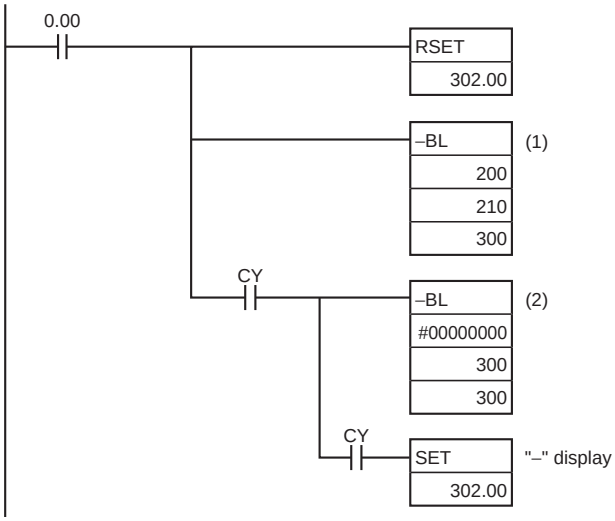
Note **10's Complement**

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 211 and CIO 210 is subtracted from the content of CIO 201 and CIO 200, and the result is output in eight-digit BCD to CIO 301 and CIO 300. The result is negative, so the instruction at (2) will be executed, and the true value will then be output to CIO 301 and CIO 300.



Subtraction at 1

Mi+1: CIO 201 Mi: CIO 200

0	9	5	8

3	9	6	0

Su+1: CIO 211 Su: CIO 210

– | 1 | 7 | 0 | 7 |

2	6	4	1

09583960 + (100000000 – 17072641)

CY R+1: CIO 301 R+1: CIO 300

1

9	2	5	1

1	3	1	9

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: CIO 301 Su: CIO 300

– | 9 | 2 | 5 | 1 |

1	3	1	9

00000000 + (100000000 – 92511319)

CY R+1: CIO 301 R+1: CIO 300

1

0	7	4	8

8	6	8	1

Final Subtraction Result

Mi+1: CIO 201 Mi: CIO 200

2	0	F	5

5	A	1	0

Su+1: CIO 211 Su: CIO 210

– | 6 | 8 | 5 | 1 |

F	9	2	D

CY R+1: CIO 301 R+1: CIO 300

1

0	7	4	8

8	6	8	1

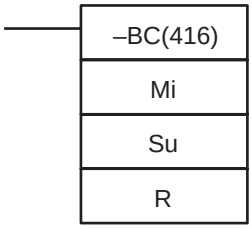
The Carry Flag (CY) will be turned ON, so the actual number is –7,488,681. Because the content of CIO 301 and CIO 300 is negative, CY is used to turn ON CIO 302.00 to indicate this.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

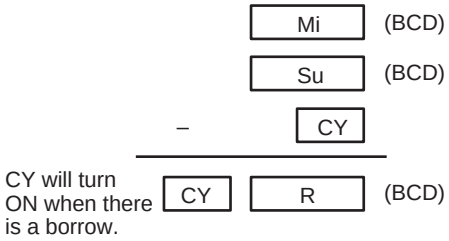
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #9999 (BCD)		---
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

–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 2's complement.



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 0. 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.

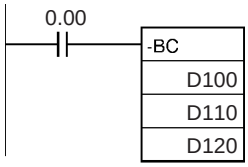
If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a borrow, the Carry Flag will turn ON.

Note To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0.00 is ON in the following example, D110 and CY will be subtracted from D100 as 4-digit BCD values, and the result will be output to D120.

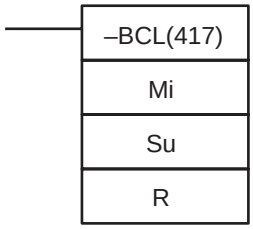


3-10-16 DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

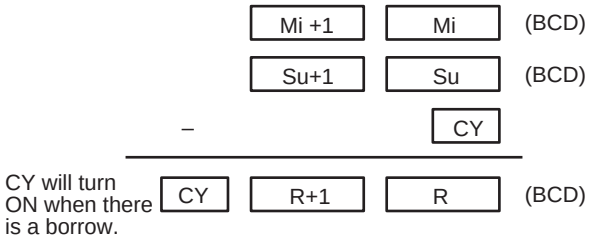
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #99999999 (BCD)		---
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 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 0. 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.

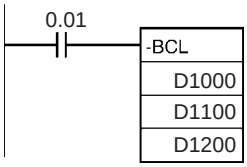
If as a result of the subtraction, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

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 0.01 is ON in the following example, D1101, D1100, and CY will be subtracted from D1001 and D1000 as 8-digit BCD values, and the result will be output to D1201 and D1200.



If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), 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, as an input condition of the Carry Flag (CY). The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

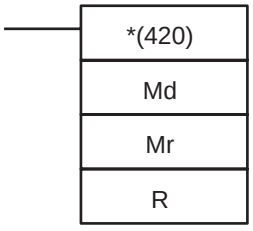
Note **10's Complement**
A 10's complement is the value obtained by subtracting each digit from 9 and adding one to the result. For example, the 10's complement for 7556 is calculated as follows: 9999 – 7556 + 1 = 2444. For a four digit number, the 10's complement of A is 9999 – A + 1 = B. To obtain 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: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*(420)
	Executed Once for Upward Differentiation	@*(420)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

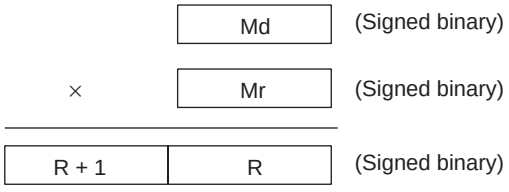
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 32767 (signed decimal)		---
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

*(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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

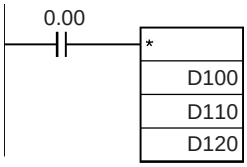
When *(420) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R is 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 0.00 is ON in the following example, D100 and D110 will be multiplied as 4-digit signed hexadecimal values and the result will be output to D121 and D120.



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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

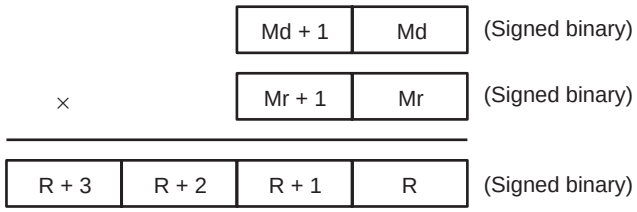
Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0 to CIO 6142		CIO 0 to CIO 6140
Work Area	W0 to W510		W0 to W508
Holding Bit Area	H0 to H510		H0 to H508
Auxiliary Bit Area	A0 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D0 to D32766		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal) –2147483648 to 0 to 2147483647 (signed decimal)		---
Data Registers	---		

Area	Md	Mr	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*L(421) multiplies the signed binary values in Md and Md+1 and 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. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

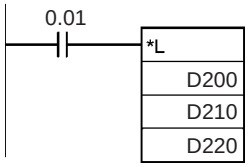
When *L(421) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 0000 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 0.01 is ON in the following example, D201, D200 and D211, D210 will be multiplied as 8-digit signed hexadecimal values and the result will be output to D220 to D223.

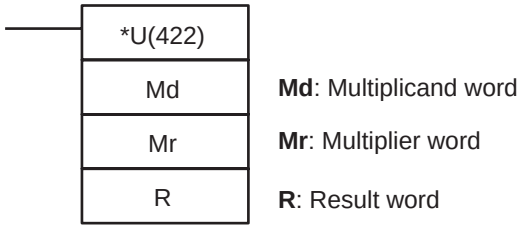


3-10-19 UNSIGNED BINARY MULTIPLY: *U(422)

Purpose

Multiplies 4-digit unsigned hexadecimal data and/or constants.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	*U(422)
	Executed Once for Upward Differentiation	@*U(422)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal)		---
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

*U(420) multiplies the 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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

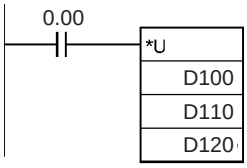
When *U(422) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1 is 0000 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 0.00 is ON in the following example, D100 and D110 will be multiplied as 4-digit unsigned binary values and the result will be output to D121 and D120.

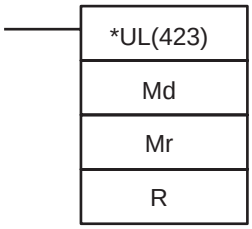


3-10-20 DOUBLE UNSIGNED BINARY MULTIPLY: *UL(423)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

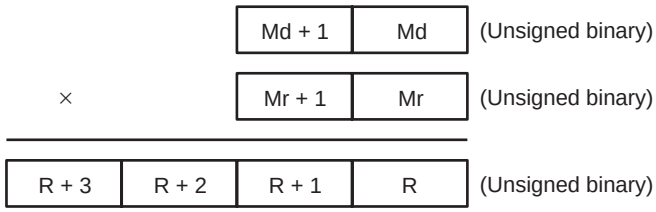
Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0 to CIO 6142		CIO 0 to CIO 6140
Work Area	W0 to W510		W0 to W508
Holding Bit Area	H0 to H510		H0 to H508
Auxiliary Bit Area	A0 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D0 to D32766		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal)		---
Data Registers	---		

Area	Md	Mr	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*UL(423) multiplies the unsigned binary values in Md and Md+1 and Mr and Mr+1 and outputs the result to R to R+3.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

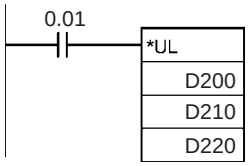
When *UL(423) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R to R+3 is 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R to R+3 is 1, the Negative Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, D201, 200, D211, and D210 will be multiplied as 8-digit unsigned binary values and the result will be output to D220 to D223.

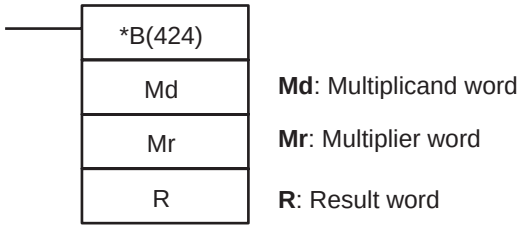


3-10-21 BCD MULTIPLY: *B(424)

Purpose

Multiplies 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Md: Multiplicand word

Mr: Multiplier word

R: 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.
Immediate Refreshing Specification		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 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #9999 (BCD)		---
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

*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 0. OFF in all other cases.

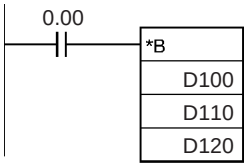
Precautions

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 hex, the Equals Flag will turn ON.

Examples

When CIO 0.00 is ON in the following example, D100 and D110 will be multiplied as 4-digit BCD values and the result will be output to D121 and D120.



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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

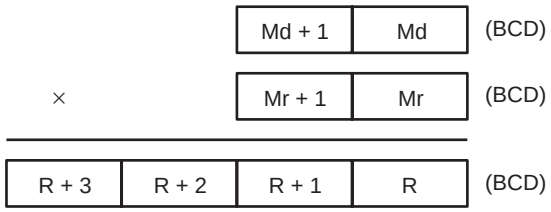
Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0 to CIO 6142		CIO 0 to CIO 6140
Work Area	W0 to W510		W0 to W508
Holding Bit Area	H0 to H510		H0 to H508
Auxiliary Bit Area	A0 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D0 to D32766		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #99999999 (BCD)		---
Data Registers	---		

Area	Md	Mr	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*BL(425) multiplies BCD values in Md and Md+1 and Mr and Mr+1 and outputs the result to R to 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. OFF in all other cases.

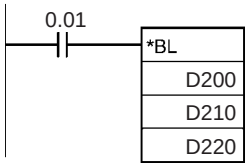
Precautions

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 00000000 hex, the Equals Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, D201, D200, D211, and D210 will be multiplied as 8-digit unsigned BCD values and the result will be output to D220 to D223.

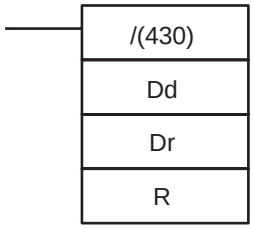


3-10-23 SIGNED BINARY DIVIDE: /(430)

Purpose

Divides 4-digit (single-word) signed hexadecimal data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	/(430)
	Executed Once for Upward Differentiation	@/(430)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

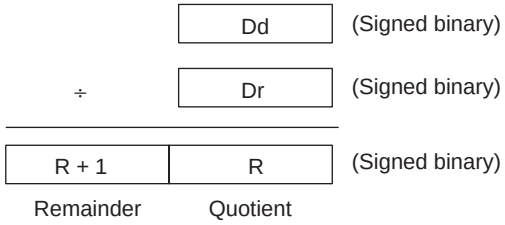
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal) –32768 to 0 to 32767 (signed decimal)	#0001 to #FFFF (binary) &1 to &65535 (unsigned decimal) –32768 to –1, 1 to 32767 (signed decimal)	---
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

/(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 the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R is 1. OFF in all other cases.

Precautions

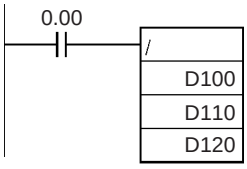
When the content of Dr is 0, an error will be generated and the Error Flag will turn ON.

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 0.00 is ON in the following example, D100 will be divided by D110 as 4-digit signed binary values, the quotient will be output to D120, and the remainder to D121.

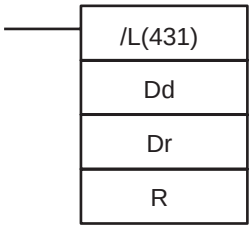


3-10-24 DOUBLE SIGNED BINARY DIVIDE: /L(431)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

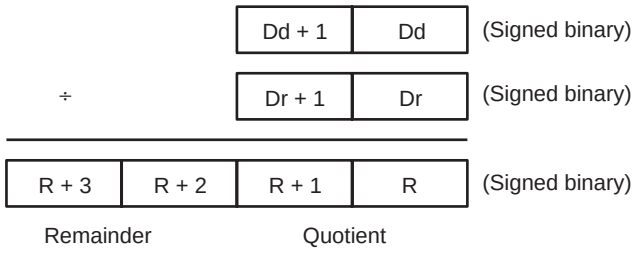
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6142		CIO 0 to CIO 6140
Work Area	W0 to W510		W0 to W508
Holding Bit Area	H0 to H510		H0 to H508

Area	Dd	Dr	R
Auxiliary Bit Area	A0 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D0 to D32766		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal) –2147483647 to 2147483647 (signed decimal)	#00000001 to #FFFFFFFF (binary) &1 to &4294967295 (unsigned decimal) –2147483648 to –1, 1 to 2147483647 (signed decimal)	---
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 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 the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R+1, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R+1, R is 1. OFF in all other cases.

Precautions

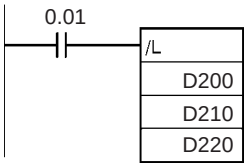
When the remainder of the result, R+3, R+2 is 0,the Error Flag will turn ON.

If as a result of the division, the content of R+1, R is 00000000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R+1, R is 1, the Negative Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, D201 and D200 will be divided by D211 and D210 as 8-digit signed hexadecimal values, the quotient will be output to D221 and D220, and the remainder will be output to D223 and D222.

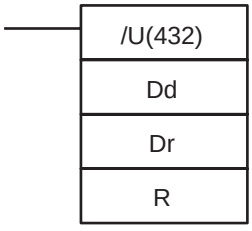


3-10-25 UNSIGNED BINARY DIVIDE: /U(432)

Purpose

Divides 4-digit (single-word) unsigned hexadecimal data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

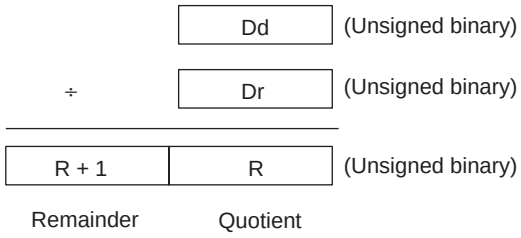
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary) &0 to &65535 (unsigned decimal)	#0001 to #FFFF (binary) &1 to &65535 (unsigned decimal)	---
Data Registers	DR0 to 15		---

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 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 the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R is 1. OFF in all other cases.

Precautions

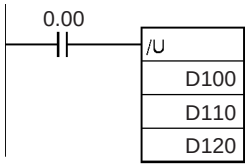
If as a result of the division, the content of R+1 is 0, the Error Flag will turn ON.

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 0.00 is ON in the following example, D100 will be divided by D110 as 4-digit unsigned binary values, the quotient will be output to D120, and the remainder will be output to D121.

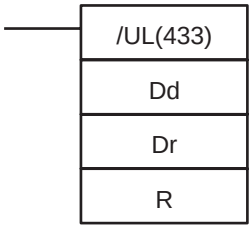


3-10-26 DOUBLE UNSIGNED BINARY DIVIDE: /UL(433)

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		
Variations	**Executed Each Cycle for ON Condition**	/UL(433)
	Executed Once for Upward Differentiation	@/UL(433)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

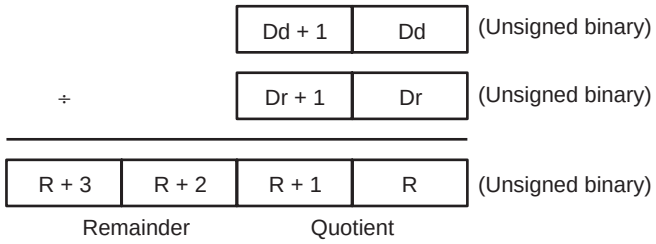
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6142		CIO 0 to CIO 6140
Work Area	W0 to W510		W0 to W508
Holding Bit Area	H0 to H510		H0 to H508
Auxiliary Bit Area	A0 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D0 to D32766		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary) &0 to &4294967295 (unsigned decimal)	#00000001 to #FFFFFFFF (binary) &1 to &4294967295 (unsigned decimal)	---
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 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 the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division R+1, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R+1, R is 1. OFF in all other cases.

Precautions

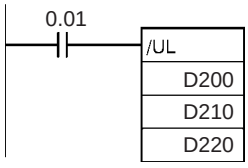
When the content of Dr, Dr+1 is 0, the Error Flag will turn ON.

If as a result of the division, the content of R, R+1, is 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 0.01 is ON in the following example, D201 and D200 will be divided by D211 and D210 as 8-digit unsigned hexadecimal values, the quotient will be output to D221 and D220, and the remainder will be output to D223 and D222.

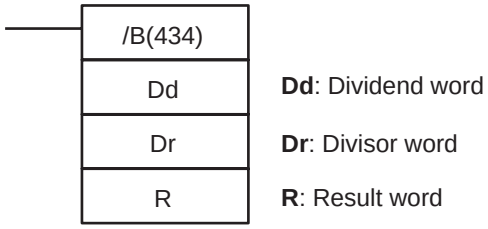


3-10-27 BCD DIVIDE: /B(434)

Purpose

Divides 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/B(434)
	Executed Once for Upward Differentiation	@/B(434)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

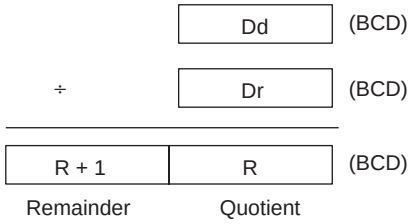
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #9999 (BCD)	#0001 to #9999 (BCD)	---
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

/B(434) divides the BCD content of Dd by those 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 the remainder is 0. OFF in all other cases.
Equals Flag	=	ON when R is 0. OFF in all other cases.

Precautions

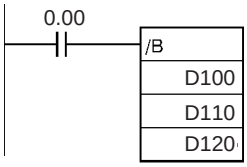
If Dd or Dr are not BCD or if the remainder (R+1) is 0, an error will be generated and the Error Flag will turn ON.

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 leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 0.00 is ON in the following example, D100 will be divided by D110 as 4-digit BCD values and the quotient will be output to D120 and the remainder to D121.

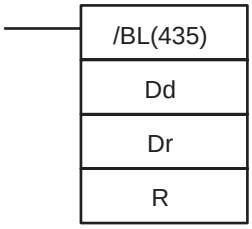


3-10-28 DOUBLE BCD DIVIDE: /BL(435)

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

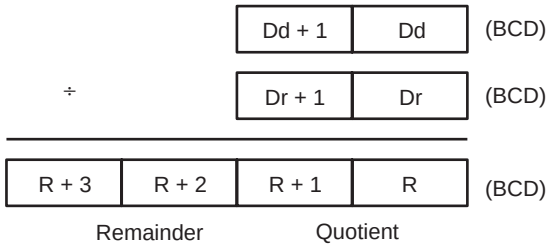
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6142		CIO 0 to CIO 6140
Work Area	W0 to W510		W0 to W508
Holding Bit Area	H0 to H510		H0 to H508
Auxiliary Bit Area	A0 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D0 to D32766		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #99999999 (BCD)	#00000001 to #99999999 (BCD)	---
Data Registers	---		

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 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. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.

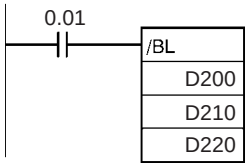
Precautions

If Dd, Dd+1 and/or Dr, Dr+1 are not BCD or the content of Dr, Dr+1 is 0, 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 00000000 hex, the Equals Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, D201 and D200 will be divided by D211 and D210 as 8-digit BCD values, the quotient will be output to D221 and D220, and the remainder will be output to D223 and D222.



3-11 Conversion Instructions

This section describes instructions used for data conversion.

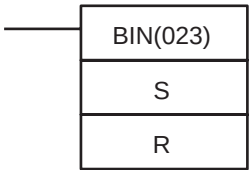
Instruction	Mnemonic	Function code	Page
BCD-TO-BINARY	BIN	023	390
DOUBLE BCD-TO-DOUBLE BINARY	BINL	058	391
BINARY-TO-BCD	BCD	024	393
DOUBLE BINARY-TO-DOUBLE BCD	BCDL	059	394
2'S COMPLEMENT	NEG	160	396
DOUBLE 2'S COMPLEMENT	NEGL	161	398
16-BIT TO 32-BIT SIGNED BINARY	SIGN	600	399
DATA DECODER	MLPX	076	401
DATA ENCODER	DMPX	077	405
ASCII CONVERT	ASC	086	409
ASCII TO HEX	HEX	162	412
COLUMN TO LINE	LINE	063	416
LINE TO COLUMN	COLM	064	418
SIGNED BCD-TO-BINARY	BINS	470	420
DOUBLE SIGNED BCD-TO-BINARY	BISL	472	423
SIGNED BINARY-TO-BCD	BCDS	471	426
DOUBLE SIGNED BINARY-TO-BCD	BDSL	473	428
GRAY CODE CONVERSION	GRY	474	431

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
Immediate Refreshing Specification		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 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	

Area	S	R
Auxiliary Bit Area	A0 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

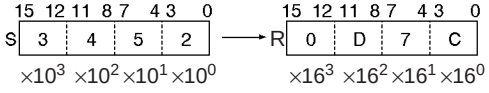
BIN(023) converts the BCD data in S to binary data and writes the result to R.

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.

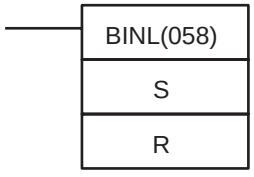


3-11-2 DOUBLE BCD-TO-DOUBLE BINARY: BINL(058)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

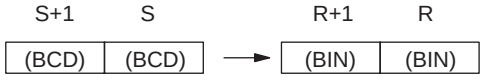
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.

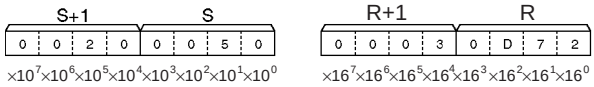


Flags

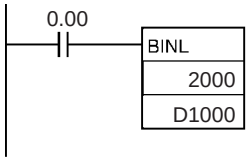
Name	Label	Operation
Error Flag	ER	ON if the contents of S+1, S are not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	OFF

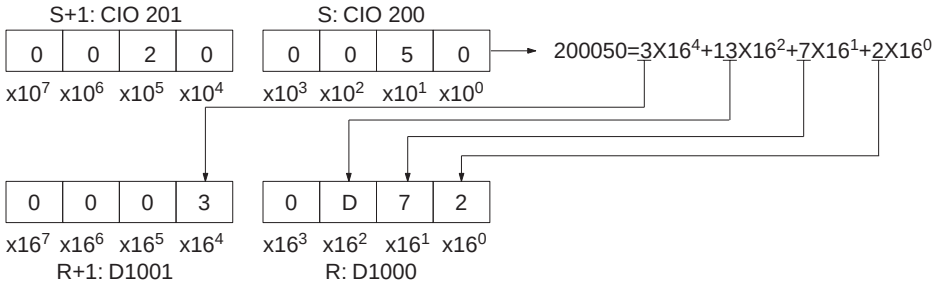
Examples

The following diagram shows an example of 8-digit BCD-to-binary conversion.



When CIO 0.00 is ON in the following example, the 8-digit BCD value in CIO 201 and CIO 200 is converted to hexadecimal and stored in D1001 and D1000.



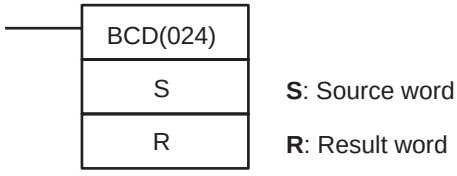


3-11-3 BINARY-TO-BCD: BCD(024)

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
Immediate Refreshing Specification		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 decimal).

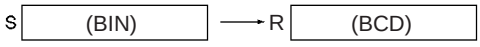
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	A448 to A959

Area	S	R
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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

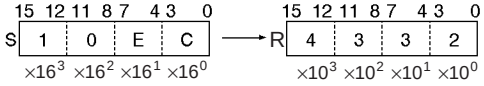
Name	Label	Operation
Error Flag	ER	ON if the content of S exceeds 270F (9999 decimal). OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

The content of S must not exceed 270F (9999 decimal).

Example

The following diagram shows an example BCD-to-binary conversion.

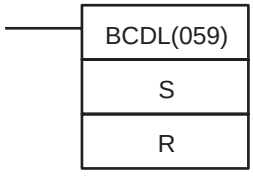


3-11-4 DOUBLE BINARY-TO-DOUBLE BCD: BCDL(059)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word

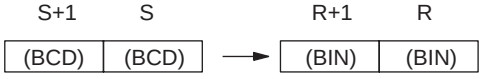
The content of S+1 and S must be between 0000 0000 and 05F5 E0FF hexadecimal (0000 0000 and 9999 9999 decimal).

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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

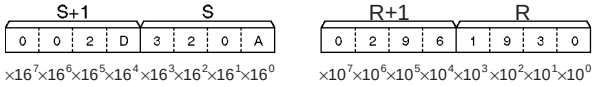
Name	Label	Operation
Error Flag	ER	ON if the contents of S and S+1 exceed 05F5 E0FF (9999 9999 decimal). OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.

Precautions

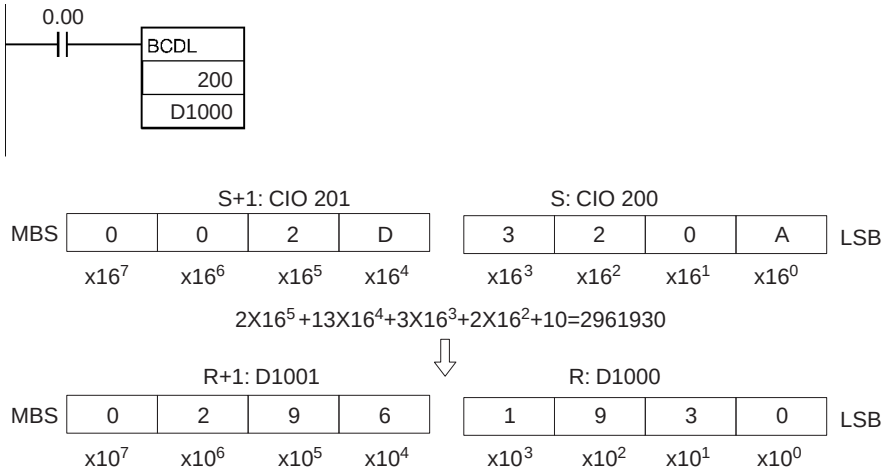
The content of S+1 and S must not exceed 05F5 E0FF (9999 9999 decimal).

Examples

The following diagram shows an example of 8-digit BCD-to-binary conversion.



When CIO 0.00 is ON in the following example, the hexadecimal value in CIO 201 and CIO 200 is converted to a BCD value and stored in D1001 and D1000.

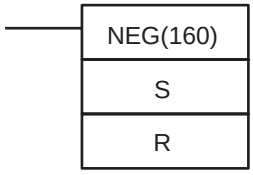


3-11-5 2'S COMPLEMENT: NEG(160)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

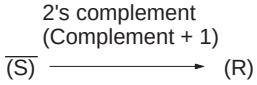
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	

Area	S	R
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
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 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.

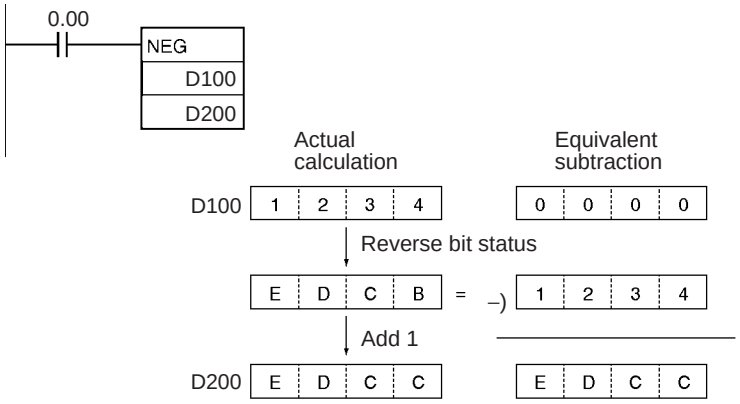
Flags

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 0.00 is ON in the following example, NEG(160) calculates the 2's complement of the content of D100 and writes the result to D200.

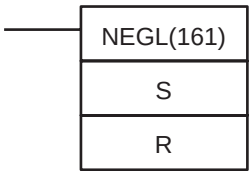


3-11-6 DOUBLE 2'S COMPLEMENT: NEGL(161)

Purpose

Calculates the 2's complement of two words of hexadecimal data.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	NEGL(161)
	Executed Once for Upward Differentiation	@NEGL(161)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

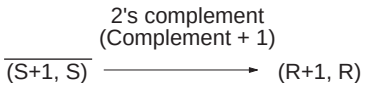
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

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.



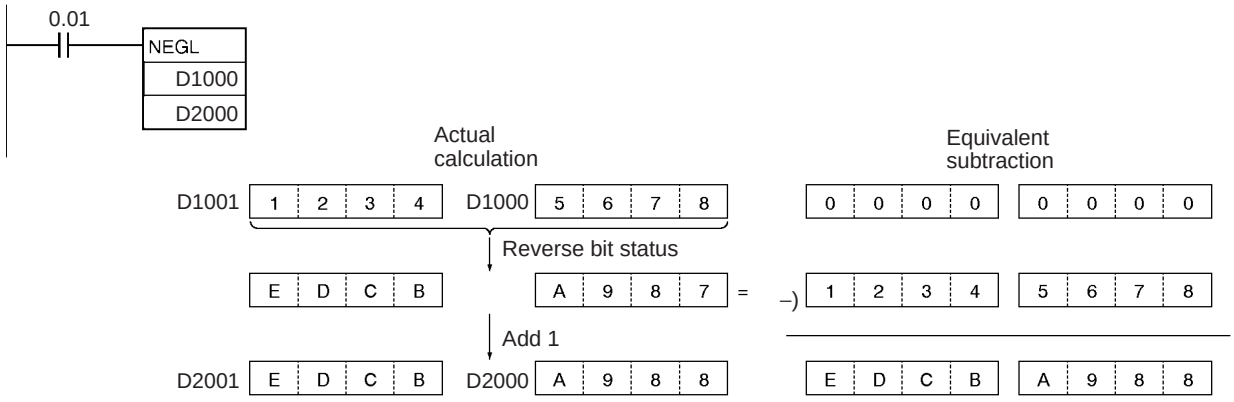
Note This operation (reversing the status of the bits and adding 1) is equivalent to subtracting the content of S+1 and S from 0000 0000.

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 hex will be 8000 hex.

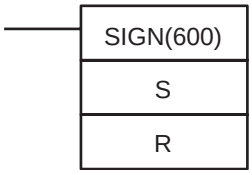
Example When CIO 0.01 is ON in the following example, NEGL(161) calculates the 2's complement of the content of D1001 and D1000 and writes the result to D2001 and D2000.



3-11-7 16-BIT TO 32-BIT SIGNED BINARY: SIGN(600)

Purpose Expands a 16-bit signed binary value to its 32-bit equivalent.

Ladder Symbol



S: Source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SIGN(600)
	Executed Once for Upward Differentiation	@SIGN(600)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		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 0 to CIO 6143	CIO 0 to CIO 6142
Work Area	W0 to W511	W0 to W510
Holding Bit Area	H0 to H511	H0 to H510
Auxiliary Bit Area	A0 to A959	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094

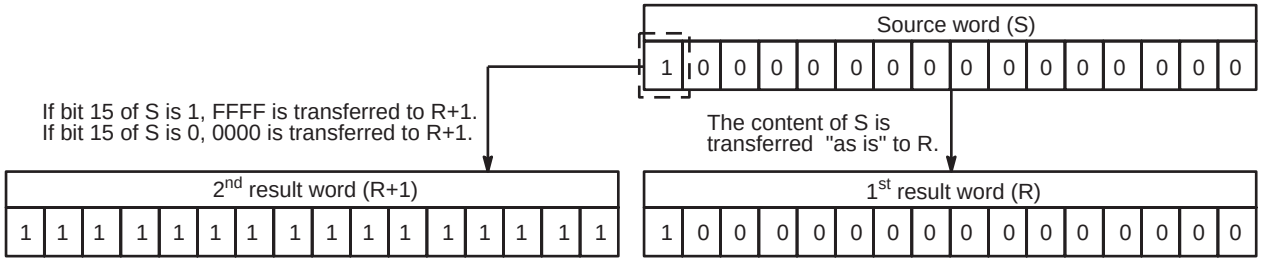
Area	S	R
Counter Area	C0000 to C4095	C0000 to C4094
DM Area	D0 to D32767	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to 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.

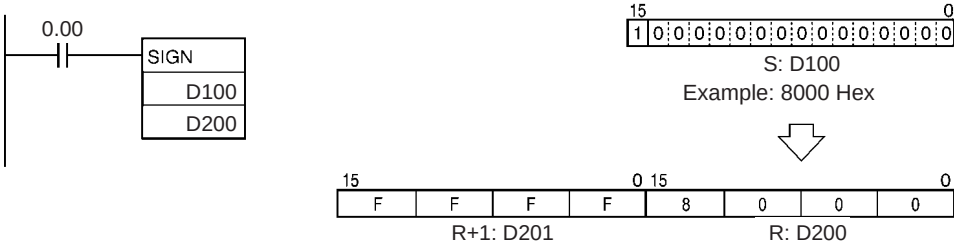


Flags

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 0.00 is ON in the following example, SIGN(600) converts the 16-bit signed binary content of D100 (#8000 = –32,768 decimal) to its 32-bit equivalent (#FFFF 8000 = –32,768 decimal) and writes that result to D201 and D200.

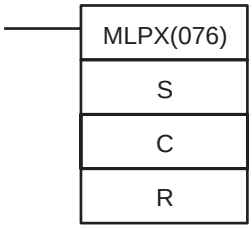


3-11-8 DATA DECODER: MLPX(076)

Purpose

Reads the numerical value in the specified digit (or byte) in the source word, turns ON the corresponding bit in the result word (or 16-word range), and turns OFF all other bits in the result word (or 16-word range).

Ladder Symbol



S: Source word

C: Control word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	MLPX(076)
	Executed Once for Upward Differentiation	@MLPX(076)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

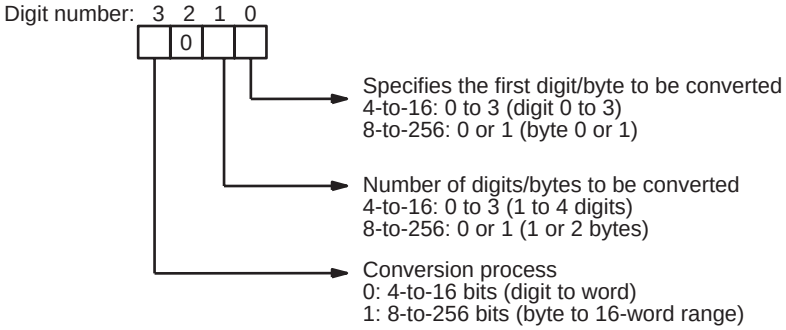
Operands

S: Source Word

The data in the source word indicates the location of the bit(s) that will be turned ON.

C: Control Word

The control word specifies whether MLPX(076) will perform a 4-to-16 bit conversion or an 8-to-256 bit conversion, the number of digits or bytes to be converted, and the starting digit or byte.



R: First result word

There can be anywhere from 1 to 32 result words, depending upon the type of conversion process and number of digits/bytes being converted. The result words must be in the same data area.

Operand Specifications

Area	S	C	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959

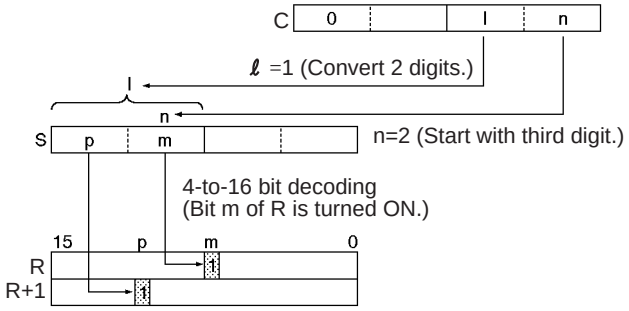
Area	S	C	R
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing 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

MLPX(076) can perform 4-to-16 bit or 8-to-256 bit conversions. Set the leftmost digit of C to 0 to specify 4-to-16 bit conversion and set it to 1 to specify 8-to-256 bit conversion.

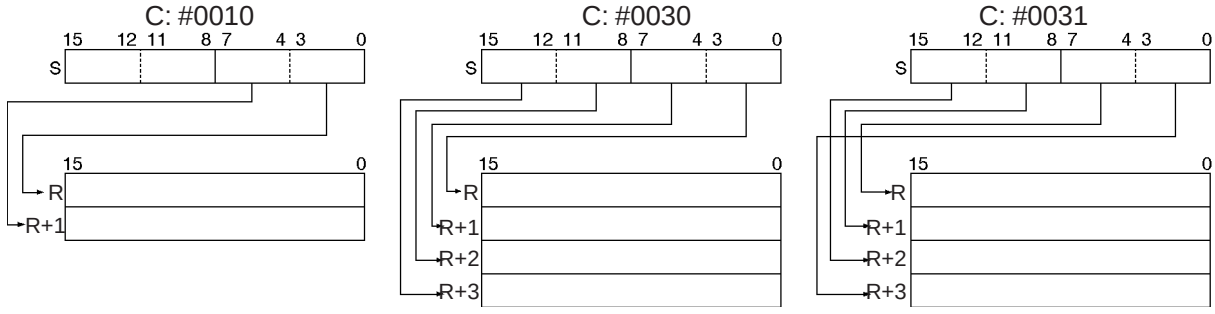
4-to-16 bit Conversion

When the leftmost digit of C is 0, MLPX(076) takes the value of the specified digit in S (0 to F) and turns ON the corresponding bit in the result word. All other bits in the result word will be turned OFF. Up to four digits can be converted.



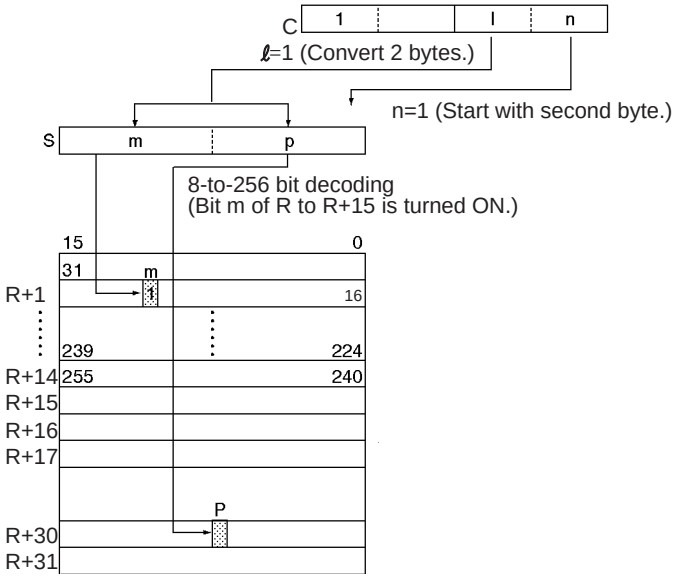
When two or more digits are being converted, MLPX(076) will read the digits in S from right to left and will wrap around to the rightmost digit after the leftmost digit, if necessary.

The following diagram shows some example values for C and the 4-to-16 bit conversions that they produce.



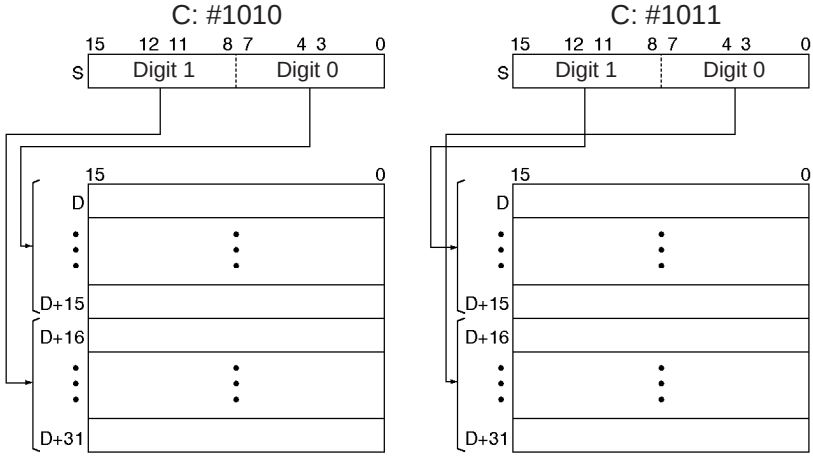
8-to-256 bit Conversion

When the leftmost digit of C is 1, MLPX(076) takes the value of the specified byte in S (00 to FF) and turns ON the corresponding bit in the range of 16 result words. All other bits in the result words will be turned OFF. Up to two bytes can be converted.



When two bytes are being converted, MLPX(076) will read the bytes in S from right to left and will wrap around to the rightmost byte if the leftmost byte (byte 1) has been specified as the starting byte.

The following diagram shows some example values for C and the 8-to-256 bit conversions that they produce.



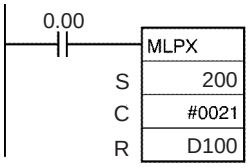
Flags

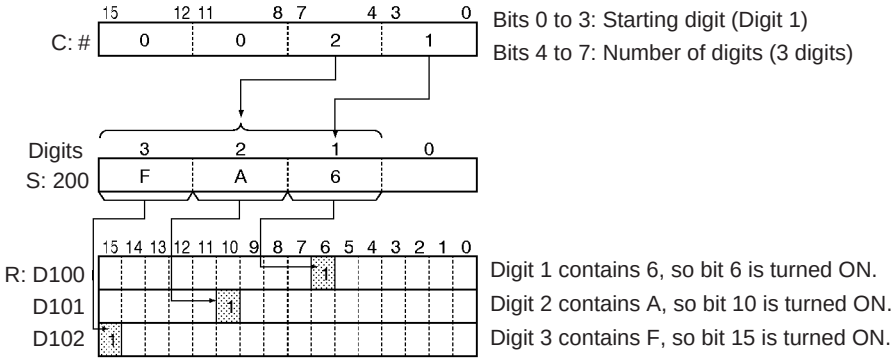
Name	Label	Operation
Error Flag	ER	ON if C is not within the specified ranges. OFF in all other cases.

Examples

4-to-16 bit Conversion

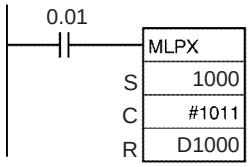
When CIO 0.00 is ON in the following example, MLPX(076) will convert 3 digits in CIO 200 beginning the second digit, as indicated by C (#0021). The corresponding bits in D100 to D102 will be turned ON.

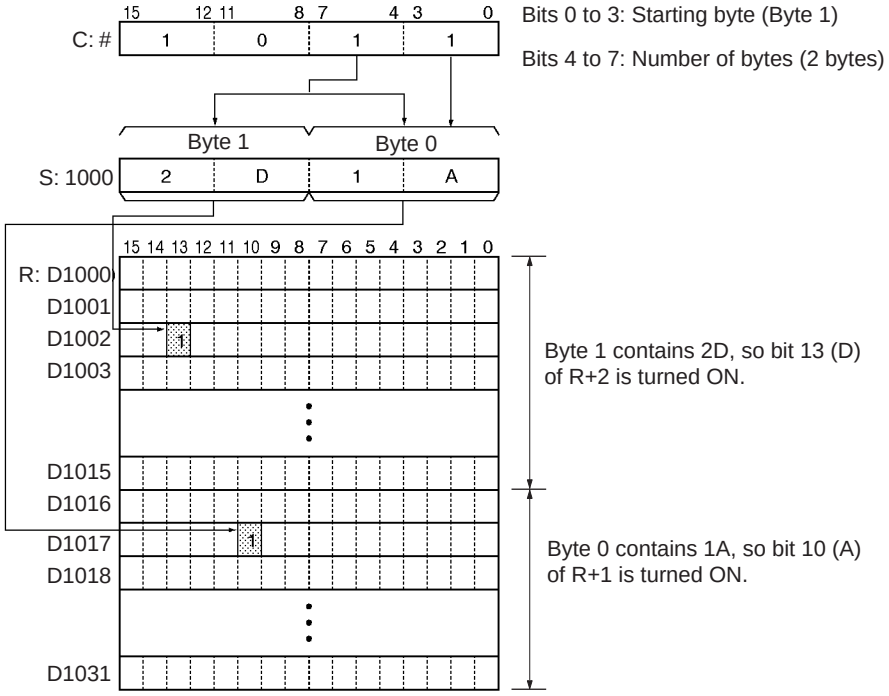




8-to-256 bit Conversion

When CIO 0.01 is ON in the following example, MLPX(076) will convert the 2 bytes in S beginning with byte 1 (the leftmost byte), as indicated by C (#1011). The corresponding bits in D1000 to D1015 and D1016 to D1031 will be turned ON.



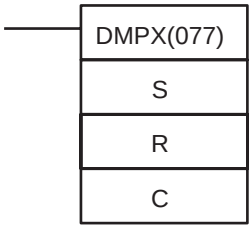


3-11-9 DATA ENCODER: DMPX(077)

Purpose

FInds the location of the first or last ON bit within the source word (or 16-word range), and writes that value to the specified digit (or byte) in the result word.

Ladder Symbol



S: First source word

R: Result word

C: Control word

Variations

Variations	**Executed Each Cycle for ON Condition**	DMPX(077)
	Executed Once for Upward Differentiation	@DMPX(077)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word

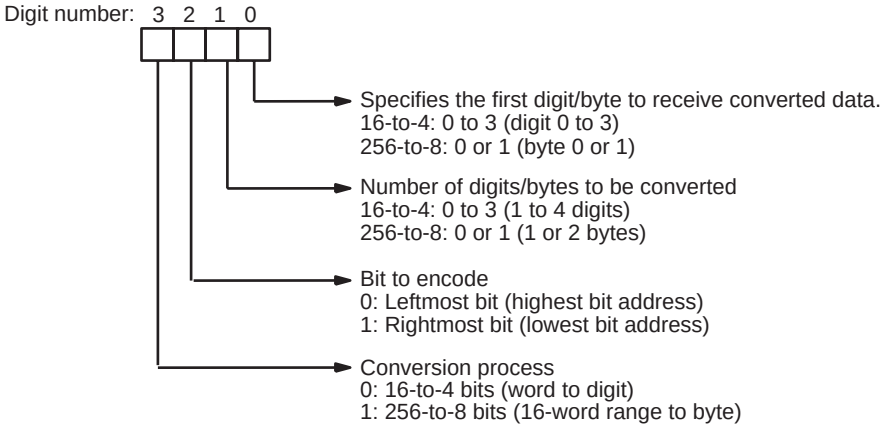
There can be anywhere from 1 to 32 source words, depending upon the type of conversion process and number of digits/bytes being converted. The source words must be in the same data area.

R: Result Word

The locations of the bits that were ON in the source word(s) are written to the digits/bytes in R starting with the specified first digit/byte.

C: Control Word

The control word specifies whether DMPX(077) will perform a 16-to-4 bit conversion or an 256-to-8 bit conversion, whether the leftmost or rightmost ON bit will be encoded, the number of digits or bytes that will be converted, and the starting digit or byte where the results will be written.



Operand Specifications

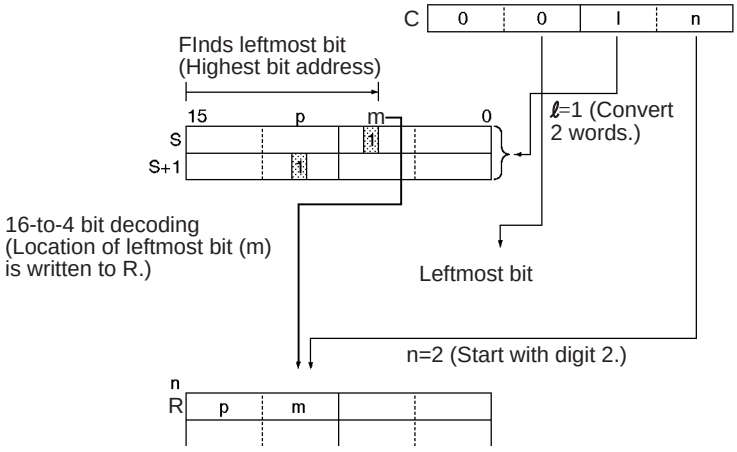
Area	S	R	C
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	A0 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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

DMPX(077) can perform 16-to-4 bit or 256-to-8 bit conversions. Set the leftmost digit of C to 0 to specify 16-to-4 bit conversion and set it to 1 to specify 256-to-8 bit conversion.

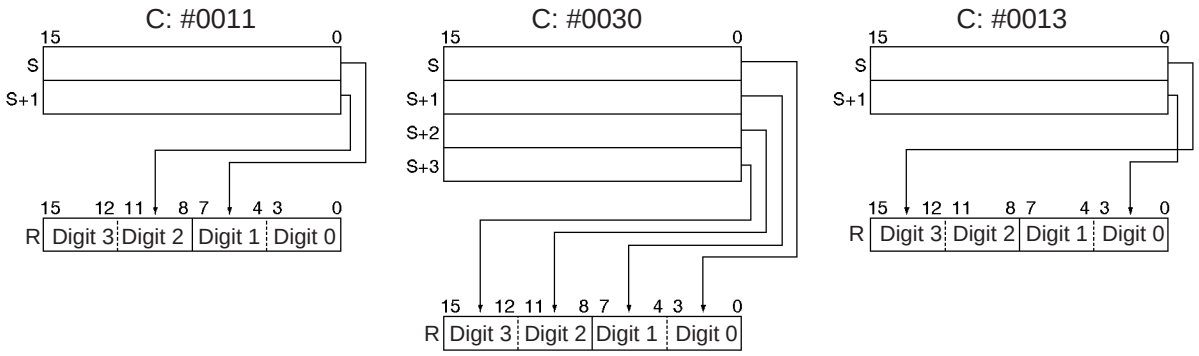
16-to-4 bit Conversion

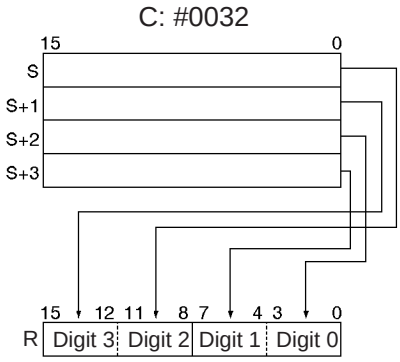
When the fourth (leftmost) digit of C is 0, DMPX(077) finds the locations of the leftmost or rightmost ON bits in up to 4 source words and writes these locations to R beginning with the specified digit. (Set the third digit of C to 0 to find the leftmost ON bits or 1 to find the rightmost ON bits.)



When two or more digits are being converted, DMPX(077) will write the values to the digits in R from right to left and will wrap around to the rightmost digit after the leftmost digit, if necessary.

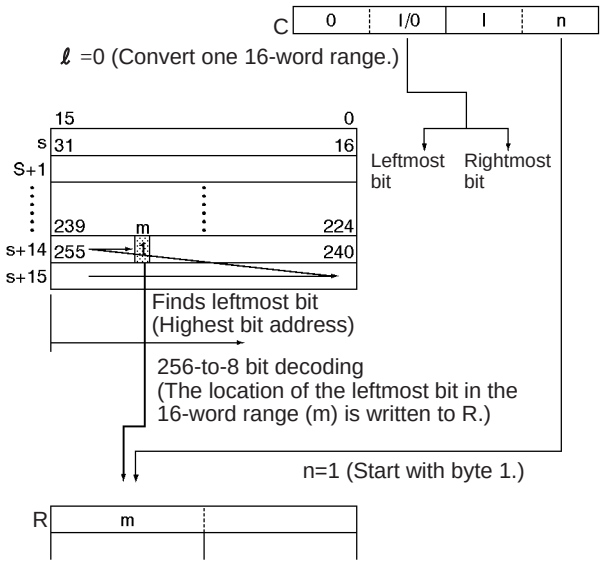
The following diagram shows some example values for C and the 16-to-4 bit conversions that they produce.





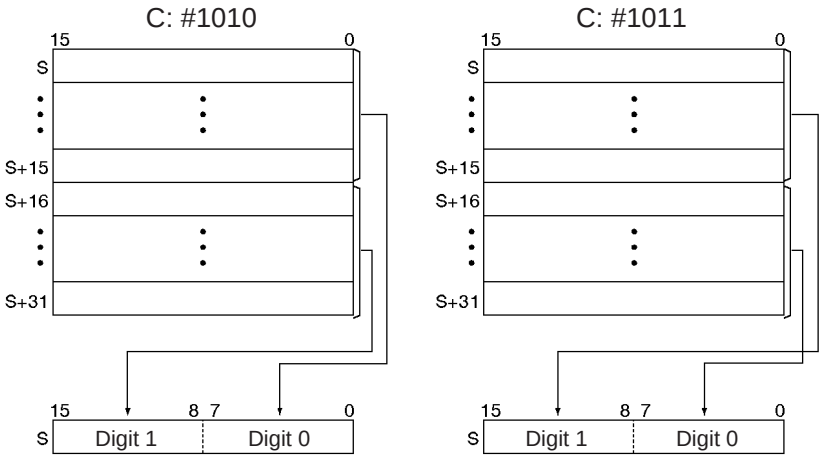
256-to-8 bit Conversion

When the fourth (leftmost) digit of C is 1, DMPX(077) finds the locations of the leftmost (highest bit address) or rightmost (lowest bit address) ON bits in one or two 16-word ranges of source words. The locations of these bits are written to R beginning with the specified byte. (Set the third digit of C to 0 to find the leftmost ON bits or 1 to find the rightmost ON bits.)



When two bytes are being converted, DMPX(077) will write the values to the bytes in R from right to left and will wrap around to the rightmost byte if the leftmost byte (byte 1) has been specified as the starting byte.

The following diagram shows some example values for C and the 256-to-8 bit conversions that they produce.



Flags

Name	Label	Operation
Error Flag	ER	ON if any of the source words contains 0000 hex (i.e., no bit to encode). ON if C is not within the specified ranges. OFF in all other cases.

Precautions

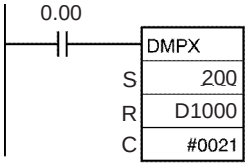
If the conversion data contains 0000 hex, but other data is to be encoded, separate the conversion by using more than one DMPX(077) instructions.

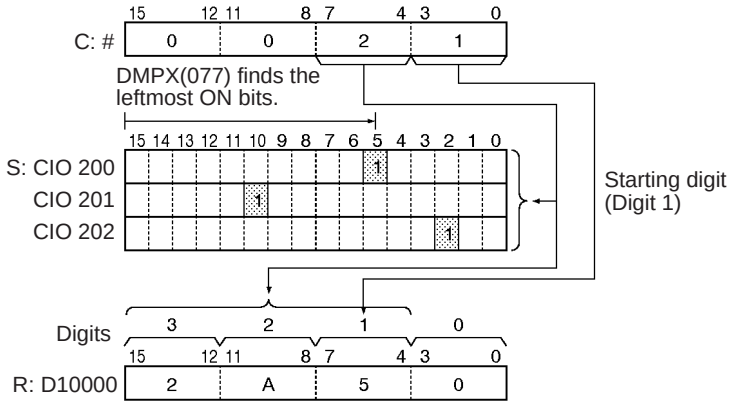
```
DMPX(077)  D0     D100   #0300
```

```
DMPX(077)  D0     D100   #0000
DMPX(077)  D1     D100   #0001
DMPX(077)  D2     D100   #0002
DMPX(077)  D3     D100   #0003
```

Examples

When CIO 0.00 is ON in the following example, DMPX(077) will find the leftmost ON bits in CIO 200 to C202 and write those locations to 3 digits in R beginning with the second digit, as indicated by C (#0021).



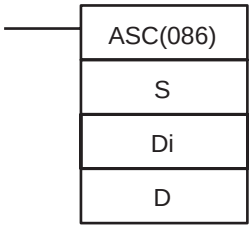


3-11-10 ASCII CONVERT: ASC(086)

Purpose

Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents.

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

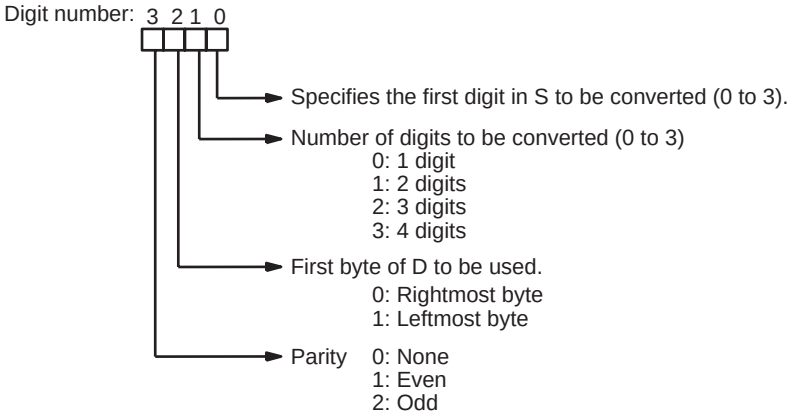
Operands

S: Source Word

Up to four digits in the source word can be converted. The digits are numbered 0 to 3, right to left.

Di: Digit Designator

The digit designator specifies various parameters for the conversion, as shown in the following diagram.



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.

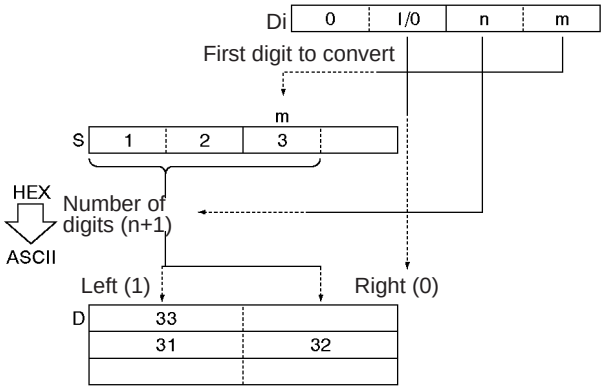
Any bytes in the destination word(s) that are not overwritten with ASCII data will be left unchanged.

Operand Specifications

Area	S	Di	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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

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.

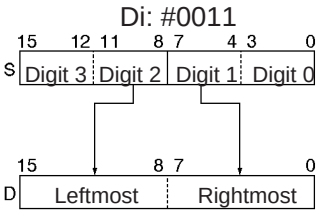
When no parity (0) is designated, the leftmost bit will always be zero. When even parity (1) is designated, the leftmost bit will be adjusted so that the total number of ON bits is even. When odd parity (2) is designated, the leftmost bit of each ASCII character will be adjusted so that there is an odd number of ON bits. The status of the parity bit does not affect the meaning of the ASCII code.

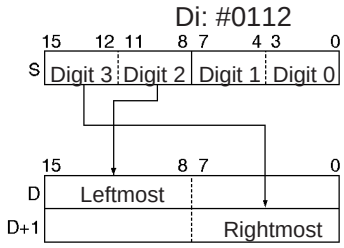
Examples of even parity:
When adjusted for even parity, ASCII “31” (00110001) will be “B1” (10110001: parity bit turned ON to create an even number of ON bits); ASCII “36” (00110110) will be “36” (00110110: parity bit remains OFF because the number of ON bits is already even).

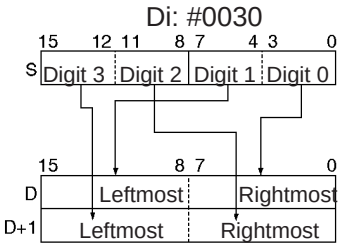
Examples of odd parity:
When adjusted for odd parity, ASCII “36” (00110110) will be “B6” (10110110: parity bit turned ON to create an odd number of ON bits); ASCII “46” (01000110) will be “46” (01000110: parity bit remains OFF because the number of ON bits is already odd).

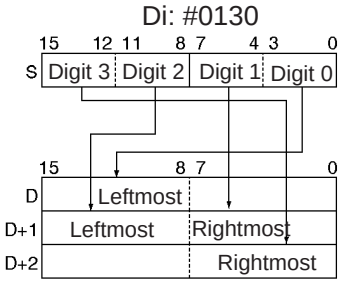
Examples of Di

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.







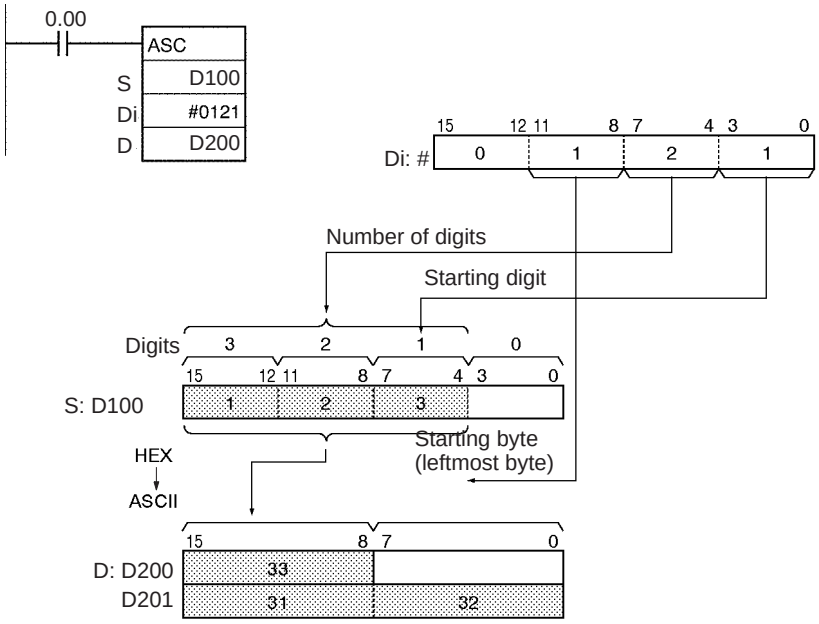


Flags

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 0.00 is ON in the following example, ASC(086) converts three hexadecimal digits in D100 (beginning with digit 1) into their ASCII equivalents and writes this data to D200 and D201 beginning with the leftmost byte in D200. In this case, a digit designator of #0121 specifies no parity, the starting byte (when writing) is the leftmost byte, the number of digits to read is 3, and the starting digit (when reading) is digit 1.

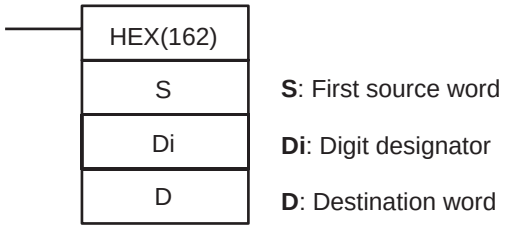


3-11-11 ASCII TO HEX: HEX(162)

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.

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

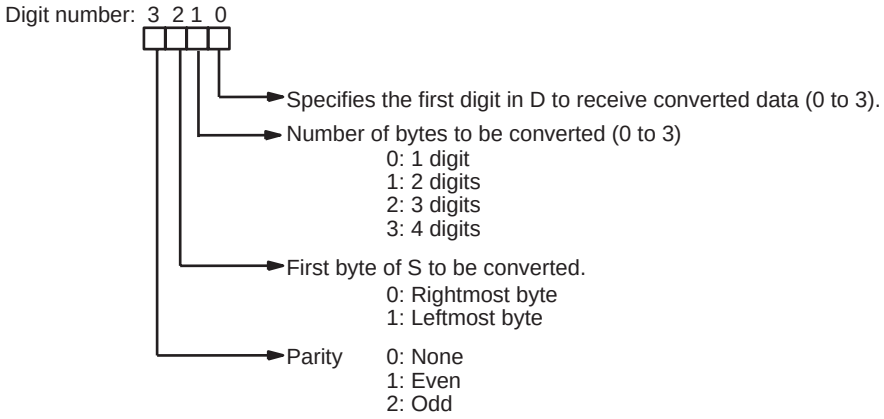
Operands

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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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

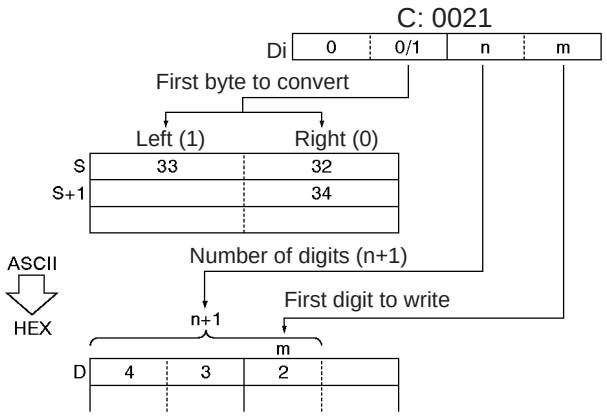
HEX(162) treats the contents of the source word(s) as ASCII data representing hexadecimal digits (0 to 9 and A to F), converts the specified number of bytes to hexadecimal, and writes the hexadecimal data to the destination word beginning at the specified digit.

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).

Flags

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

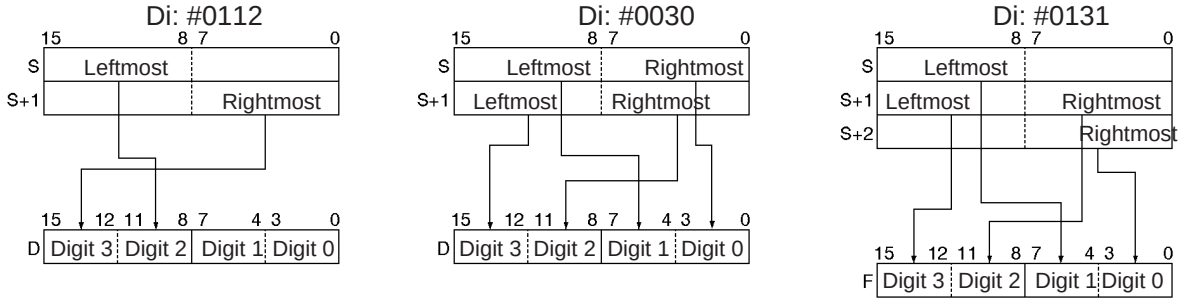
It is possible to specify the parity of the ASCII data for use in error control during data transmissions. The leftmost bit in each byte is the parity bit. With no parity the parity bit should always be zero, with even parity the status of the parity bit should result in an even number of ON bits, and 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.

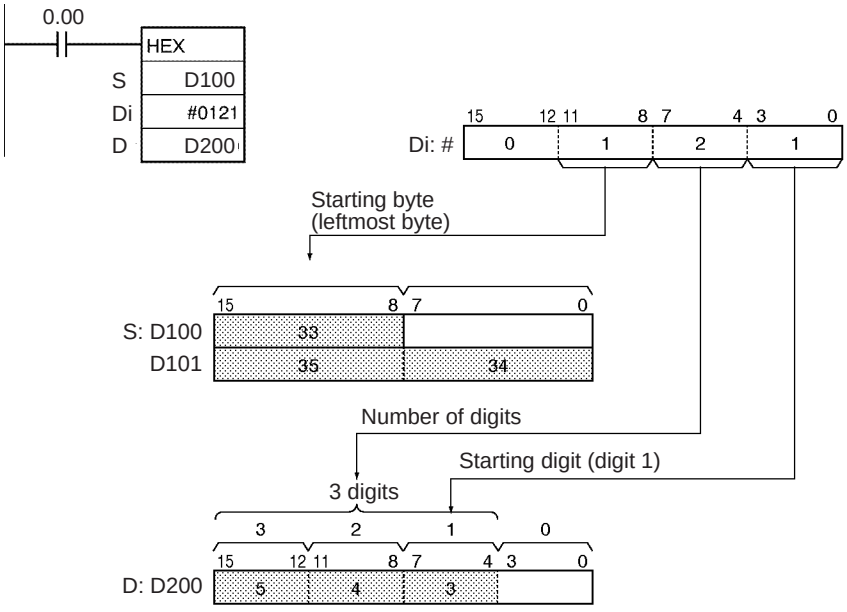
Precautions

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

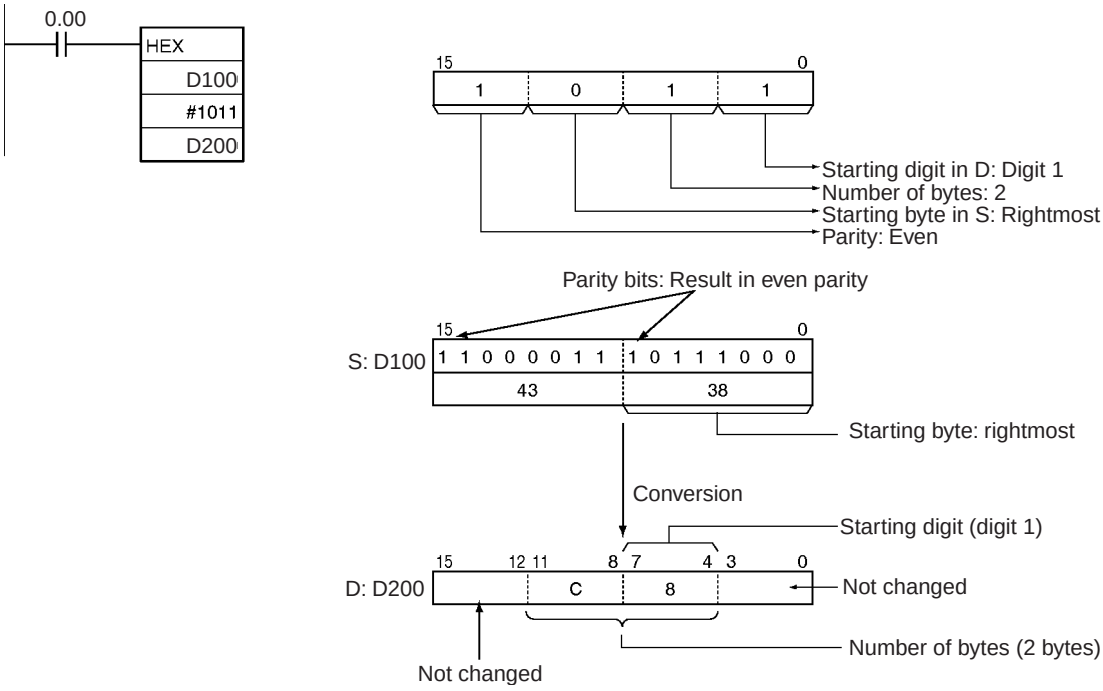
When CIO 0.00 is ON in the following example, HEX(162) converts the ASCII data in D100 and D101 according to the settings of the digit designator. (Di=#0121 specifies no parity, the starting byte (when reading) is the leftmost byte, the number of bytes to read is 3, and the starting digit (when writing) is digit 1.)

HEX(162) converts three bytes of ASCII data (3 characters) beginning with the leftmost byte of D100 into their hexadecimal equivalents and writes this data to D200 beginning with digit 1.



When CIO 0.00 is ON in the following example, HEX(162) converts the ASCII data in D100 beginning with the rightmost byte and writes the hexadecimal equivalents in D200 beginning with digit 1.

The digit designator setting of #1011 specifies even parity, the starting byte (when reading) is the rightmost byte, the number of bytes to read is 2, and the starting digit (when writing) is digit 1.)

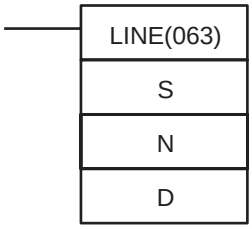


3-11-12 COLUMN TO LINE: LINE(063)

Purpose

Converts a column of bits from a 16-word range (the same bit number in 16 consecutive words) to the 16 bits of the destination word.

Ladder Symbol



S: First source word

N: Bit number

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	LINE(063)
	Executed Once for Upward Differentiation	@LINE(063)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word

Specifies the first source word. S and S+15 must be in the same data area.

N: Bit Number

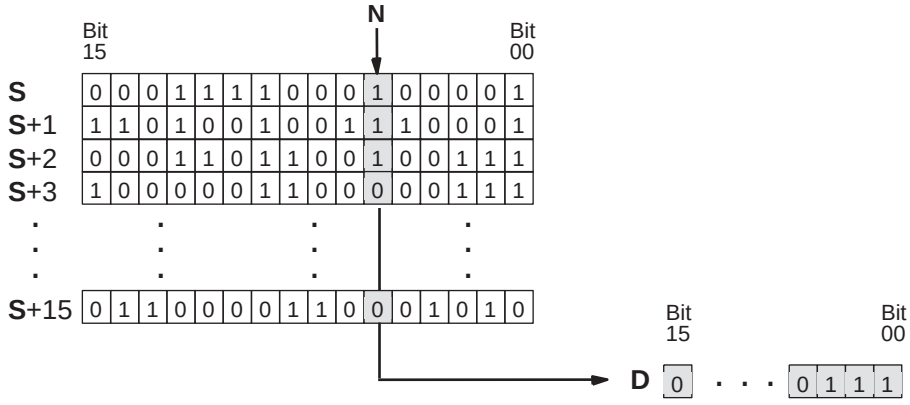
Specifies the bit number (0000 to 000F or &0 to &15) to be copied from the source words.

Operand Specifications

Area	S	N	D
CIO Area	CIO 0 to CIO 6128	CIO 0 to CIO 6143	
Work Area	W0 to W496	W0 to W511	
Holding Bit Area	H0 to H496	H0 to H511	
Auxiliary Bit Area	A0 to A944	A0 to A959	A448 to A959
Timer Area	T0000 to T4080	T0000 to T4095	
Counter Area	C0000 to C4080	C0000 to C4095	
DM Area	D0 to D32752	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

LINE(063) copies the 16 bits with bit number N from the 16-word range S to S+15 to the destination word D. Bit N of S+m is copied to bit m of D, i.e., bit N of S is copied to bit 00 of D and bit N of S+15 is copied to bit 15 of D.

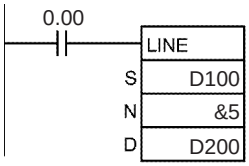


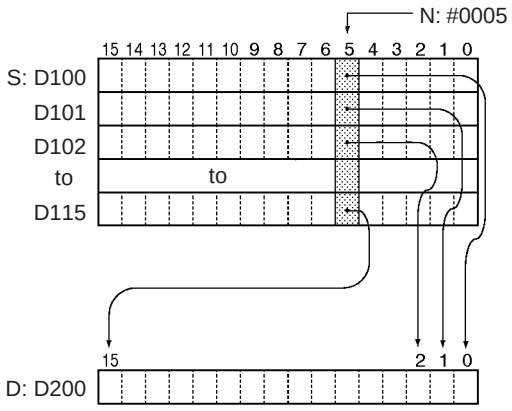
Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0000 to 000F. OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.

Example

When CIO 0.00 is ON in the following example, LINE(063) copies bit 5 from D100 to D115 to the 16 bits in D200.



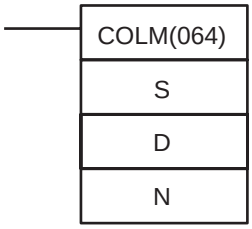


3-11-13 LINE TO COLUMN: COLM(064)

Purpose

Converts the 16 bits of the source word to a column of bits in a 16-word range of destination words (the same bit number in 16 consecutive words).

Ladder Symbol



S: Source word

D: First destination word

N: Bit number

Variations

Variations		
	Executed Each Cycle for ON Condition	COLM(064)
	Executed Once for Upward Differentiation	@COLM(064)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

D: First Destination Word

Specifies the first destination word. D and D+15 must be in the same data area.

N: Bit Number

Specifies the bit number (0000 to 000F or &0 to &15) to be overwritten by the source word.

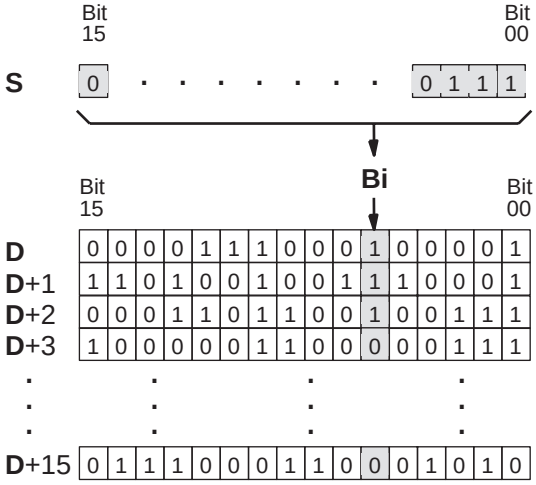
Operand Specifications

Area	S	D	N
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6128	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W496	W0 to W511

Area	S	D	N
Holding Bit Area	H0 to H511	H0 to H496	H0 to H511
Auxiliary Bit Area	A0 to A959	A448 to A944	A0 to A959
Timer Area	T0000 to T4095	T0000 to T4080	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4080	C0000 to C4095
DM Area	D0 to D32767	D0 to D32752	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	---	#0000 to #000F (binary) or &0 to &15
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

COLM(064) copies the 16 bits from S to the 16 bits with bit number N in the 16-word range D to D+15. Bit m of S is copied to bit N of D+m, i.e., bit 00 of S is copied to bit N of D and bit 15 of S is copied to bit N of D+15.

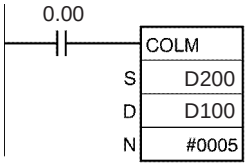


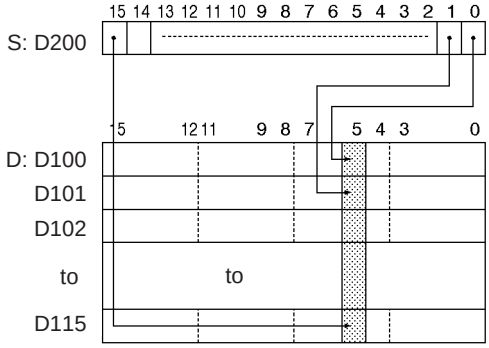
Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0000 to 000F. OFF in all other cases.
Equals Flag	=	ON if bit N is 0 in all 16 words D to D+15 after execution. OFF in all other cases.

Example

When CIO 0.00 is ON in the following example, COLM(064) copies the 16 bits in D200 (bits 00 through 15) to bit 5 in D100 through D115.



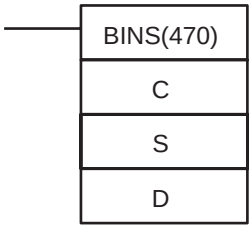


3-11-14 SIGNED BCD-TO-BINARY: BINS(470)

Purpose

Converts one word of signed BCD data to one word of signed binary data.

Ladder Symbol



C: Control word

S: Source word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	BINS(470)
	Executed Once for Upward Differentiation	@BINS(470)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

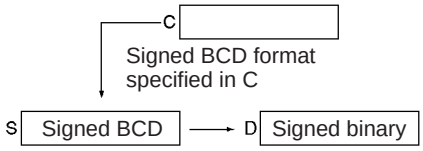
Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		

Area	C	S	D
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #0003 (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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BINS(470) converts signed BCD data to signed binary data. First the signed BCD data format and range in word S are checked against the setting in the control word (C). If the source data is correct, the signed BCD data in S is converted to signed binary and output to D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.



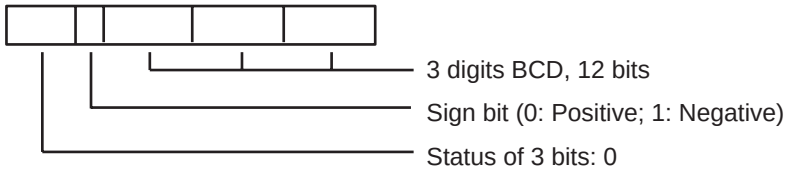
When the converted data is negative, it will be output as the 2's complement and the Negative Flag be will turned ON. NEG(160) can be used to determine the absolute value of a negative signed binary number. Refer to *3-11-5 2'S COMPLEMENT: NEG(160)396* for details.

A value of –0 in the source data will be treated as 0 and will not cause an error. Also, the status of bits 13 to 15 of S is not checked when C=0000.

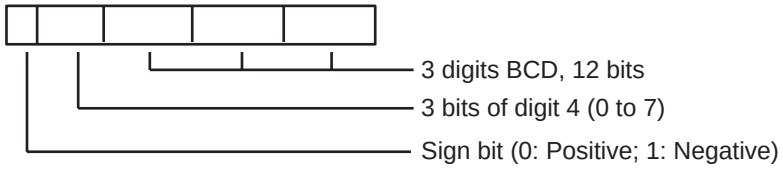
Note Some Special I/O Units output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data with BINS(470).

The control word specifies the signed BCD format as shown below.

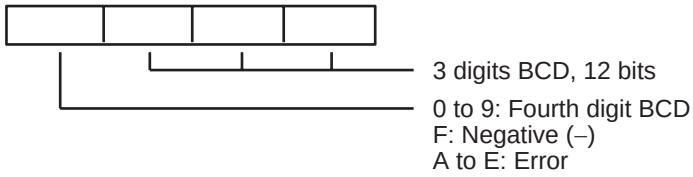
C = 0000 (Input Data Range: –999 to 999 BCD)



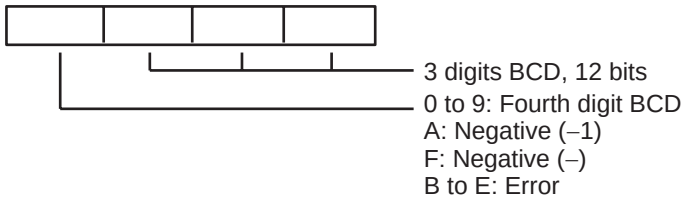
C = 0001 (Input Data Range: –7999 to 7999 BCD)



C = 0002 (Input Data Range: –999 to 9999 BCD)



C = 0003 (Input Data Range: –1999 to 9999 BCD)



The following table shows the possible BCD values for each signed BCD format and the corresponding signed binary values.

Setting	Signed BCD values	Signed binary values
C=0000	–999 to –1 and 0 to 999	FC19 to FFFF and 0000 to 03E7
C=0001	–7999 to –1 and 0 to 7999	E0C1 to FFFF and 0000 to 1F3F
C=0002	–999 to –1 and 0 to 9999	FC19 to FFFF and 0000 to 270F
C=0003	–1999 to –1 and 0 to 9999	F831 to FFFF and 0000 to 270F

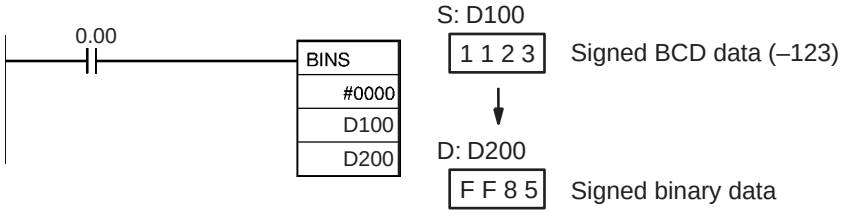
Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0002 and the leftmost digit of S is A to E. ON if C=0003 and the leftmost digit of S is B to E. ON if the content of S is not BCD. OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D is ON after execution. OFF in all other cases.

Examples

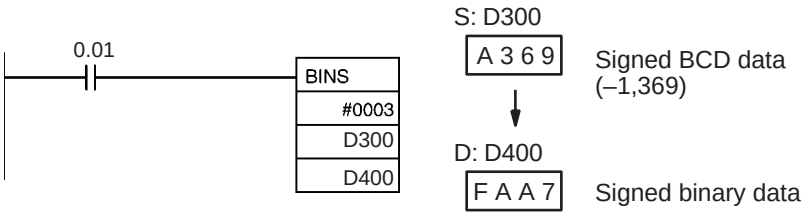
BCD Format 0 (C=#0000)

When CIO 0.00 is ON in the following example, the signed BCD data format and range in D100 are checked against the format specified in the control word (0000). The source data is correct, so the signed BCD data in D100 is converted to signed binary and output to D200.



BCD Format 0 (C=#0003)

When CIO 0.01 is ON in the following example, the signed BCD data format and range in D300 are checked against the format specified in the control word (0003). The source data is correct, so the signed BCD data in D300 is converted to signed binary and output to D400.

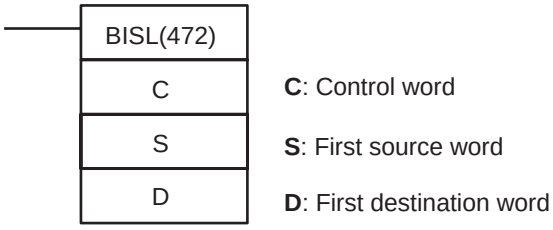


3-11-15 DOUBLE SIGNED BCD-TO-BINARY: BISL(472)

Purpose

Converts double signed BCD data to double signed binary data.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BISL(472)
	Executed Once for Upward Differentiation	@BISL(472)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

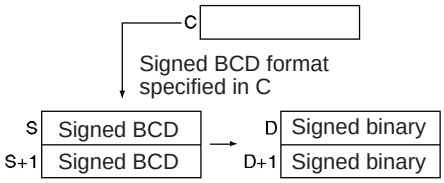
Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	
Work Area	W0 to W511	W0 to W510	
Holding Bit Area	H0 to H511	H0 to H510	
Auxiliary Bit Area	A0 to A959	A0 to A958	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094	
Counter Area	C0000 to C4095	C0000 to C4094	
DM Area	D0 to D32767	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #0003 (binary)	---	

Area	C	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BISL(472) converts the double signed BCD data in S+1 and S to double signed binary data and writes the result in D+1 and D. First the signed BCD data format and range in words S+1 and S are checked against the setting in the control word (C). If the source data is correct, the signed BCD data S+1 and S is converted to signed binary and output to D+1 and D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.



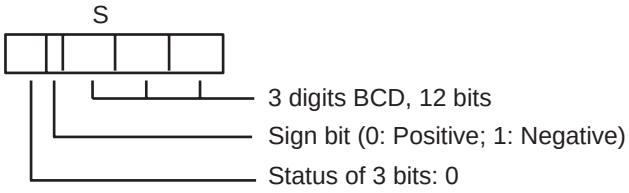
When the converted data is negative, it will be output as the 2's complement and the Negative Flag be will turned ON. NEGL(161) can be used to determine the absolute value of a negative double signed binary number. Refer to *3-11-6 DOUBLE 2'S COMPLEMENT: NEGL(161)* for details.

Values of –0 in the source data will be treated as 0 and will not cause an error. Also, the status of bits 13 to 15 of S+1 is not checked when C=0000.

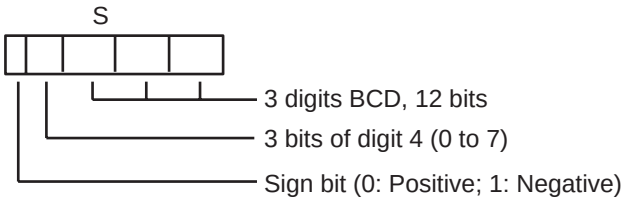
Note Some Special I/O Units output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data with BISL(472).

The control word specifies the signed BCD format as shown below.

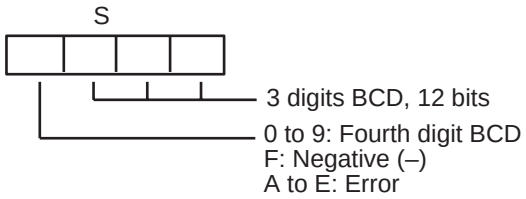
C = 0000 (Input Data Range: –999 9999 to 999 9999 BCD)



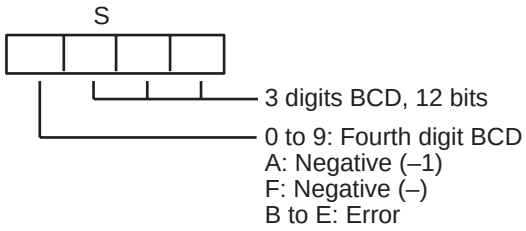
C = 0001 (Input Data Range: –7999 9999 to 7999 9999 BCD)



C = 0002 (Input Data Range: –999 9999 to 9999 9999 BCD)



C = 0003 (Input Data Range: –1999 9999 to 9999 9999 BCD)



The following table shows the possible BCD values for each signed BCD format and the corresponding signed binary values.

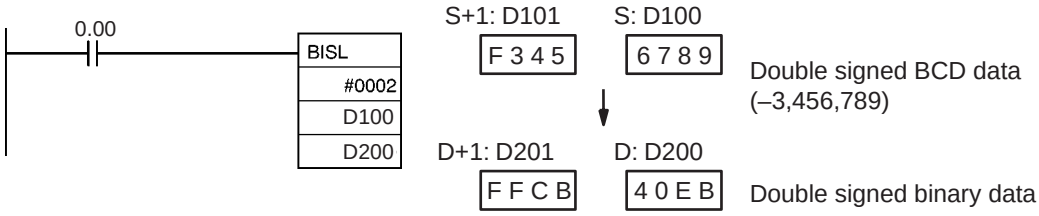
Setting	Signed BCD values	Signed binary values
C=0000	–999 9999 to –1	FF67 6981 to FFFF FFFF
	0 to 999 9999	0000 0000 to 0098 967F
C=0001	–7999 9999 to –1	FB3B 4C01 to FFFF FFFF
	0 to 7999 9999	0000 0000 to 04C4 B3FF
C=0002	–999 9999 to –1	FF67 6981 to FFFF FFFF
	0 to 9999 9999	0000 0000 to 05F5 E0FF
C=0003	–1999 9999 to –1	FECE D301 to FFFF FFFF
	0 to 9999 9999	0000 0000 to 05F5 E0FF

Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0002 and the leftmost digit of S+1 is A to E. ON if C=0003 and the leftmost digit of S+1 is B to E. ON if the content of S+1 and S is not BCD. OFF in all other cases.
Equals Flag	=	ON if D+1 contains 0000 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D+1 is ON after execution. OFF in all other cases.

Example

When CIO 0.00 is ON in the following example, the double signed BCD data format and range in D101 and D100 are checked against the format specified in the control word (0002). The source data is correct, so the double signed BCD data in D101 and D100 is converted to double signed binary and output to D201 and D200.

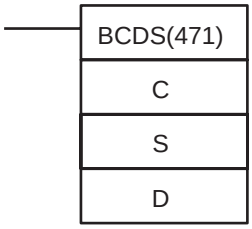


3-11-16 SIGNED BINARY-TO-BCD: BCDS(471)

Purpose

Converts one word of signed binary data to one word of signed BCD data.

Ladder Symbol



C: Control word

S: Source word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCDS(471)
	Executed Once for Upward Differentiation	@BCDS(471)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

S: Source Word

Contains the signed binary data to be converted. The content of S must be within the valid range of the BCD format specified in C.

Setting	Allowed values for S
C=0000	FC19 to FFFF or 0000 to 03E7
C=0001	E0C1 to FFFF or 0000 to 1F3F
C=0002	FC19 to FFFF or 0000 to 270F
C=0003	F831 to FFFF or 0000 to 270F

D: Destination word

Contains the converted signed BCD data. See the description section below for an explanation of the BCD formats.

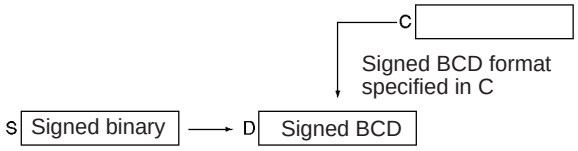
Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #0003 (binary)	---	
Data Registers	DR0 to DR15		

Area	C	S	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to 1–2048 to +2047 ,IR5 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BCDS(471) converts signed binary data to signed BCD data. First the signed binary data in word S is checked to verify that it is within the valid range for the signed BCD format specified in the control word (C). If the source data is correct, the signed binary data in S is converted to signed BCD and output to D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.



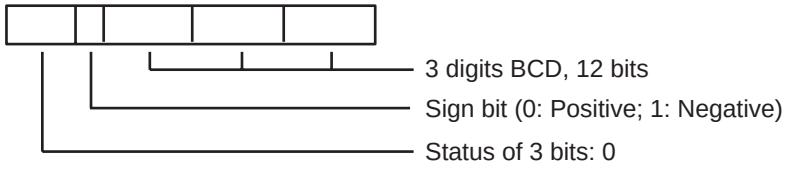
Note

(1) Values of –0 in the source data will be treated as 0 and will not cause an error.

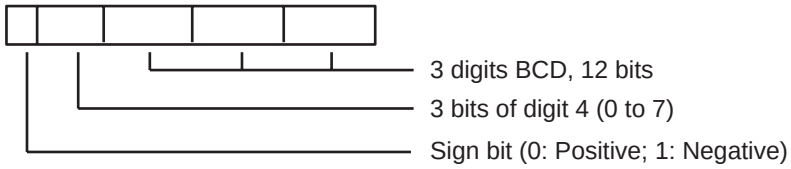
(2) Some Special I/O Units require signed BCD data inputs. BCDS(471) can be used to convert signed binary data for output to these Units.

The control word specifies the signed BCD format that will be used for the result, as shown below.

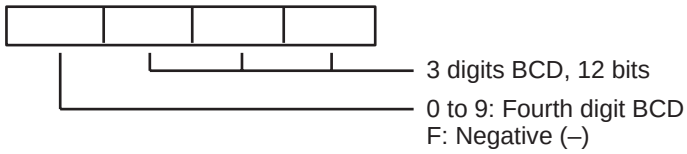
C = 0000 (Output Data Range: –999 to 999 BCD)



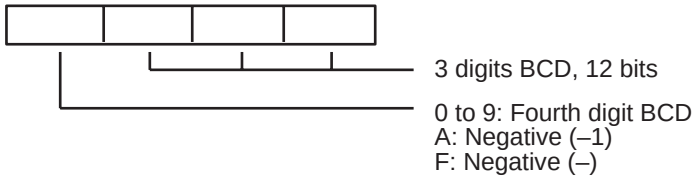
C = 0001 (Output Data Range: –7999 to 7999 BCD)



C = 0002 (Output Data Range: –999 to 9999 BCD)



C = 0003 (Output Data Range: –1999 to 9999 BCD)



The following table shows the possible signed binary values for each signed BCD format. An error will occur if the source data is not within the allowed range for the specified signed BCD format.

Setting	Signed binary values	Signed BCD values
C=0000	FC19 to FFFF and 0000 to 03E7	–999 to –1 and 0 to 999
C=0001	E0C1 to FFFF and 0000 to 1F3F	–7999 to –1 and 0 to 7999
C=0002	FC19 to FFFF and 0000 to 270F	–999 to –1 and 0 to 9999
C=0003	F831 to FFFF and 0000 to 270F	–1999 to –1 and 0 to 9999

Flags

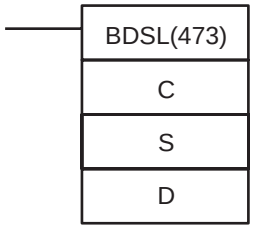
Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0000 and the source data is not within the allowed ranges (FC19 to FFFF or 0000 to 03E7). ON if C=0001 and the source data is not within the allowed ranges (E0C1 to FFFF or 0000 to 1F3F). ON if C=0002 and the source data is not within the allowed ranges (FC19 to FFFF or 0000 to 270F). ON if C=0003 and the source data is not within the allowed ranges (F831 to FFFF or 0000 to 270F). OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if C=0000 or 0001 and the result's sign bit is ON after execution. ON if C=0002 and the leftmost digit of the result is F. ON if C=0003 and the leftmost digit of the result is A or F. OFF in all other cases.

3-11-17 DOUBLE SIGNED BINARY-TO-BCD: BDSL(473)

Purpose

Converts double signed binary data to double signed BCD data.

Ladder Symbol



C: Control word

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	BDSL(473)
	Executed Once for Upward Differentiation	@BDSL(473)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Operands

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

S: First Source Word

Source words S+1 and S contain the double signed binary data to be converted. Their content must be within the valid range of the BCD format specified in C.

Setting	Allowed values for S+1 and S
C=0000	FF67 6981 to FFFF FFFF or 0000 0000 to 0098 967F
C=0001	FB3B 4C01 to FFFF FFFF or 0000 0000 to 04C4 B3FF
C=0002	FF67 6981 to FFFF FFFF or 0000 0000 to 05F5 E0FF
C=0003	FECE D301 to FFFF FFFF or 0000 0000 to 05F5 E0FF

D: First destination word

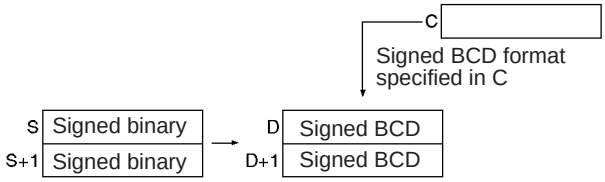
Destination words D+1 and D contain the converted double signed BCD data. See the description section below for an explanation of the BCD formats.

Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	
Work Area	W0 to W511	W0 to W510	
Holding Bit Area	H0 to H511	H0 to H510	
Auxiliary Bit Area	A0 to A959	A0 to A958	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094	
Counter Area	C0000 to C4095	C0000 to C4094	
DM Area	D0 to D32767	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #0003 (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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BDSL(473) converts double signed binary data to double signed BCD data. First the double signed binary data in S+1 and S is checked to verify that it is within the valid range for the signed BCD format specified in the control word (C). If the source data is correct, the double signed binary data in S+1 and S is converted to double signed BCD and output to D+1 and D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.



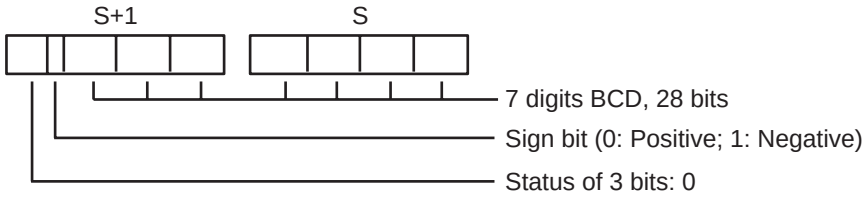
Note

(1) Values of –0 in the source data will be treated as 0 and will not cause an error.

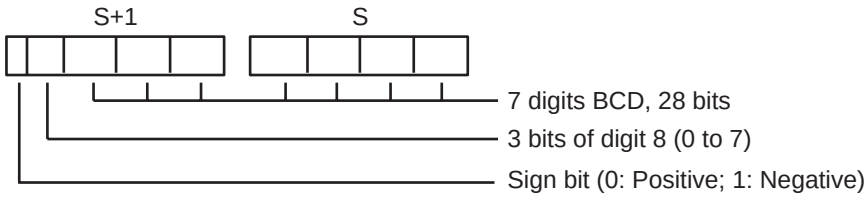
(2) Some Special I/O Units require signed BCD data inputs. BDSL(473) can be used to convert double signed binary data for output to these Units.

The control word specifies the signed BCD format that will be used for the result, as shown below.

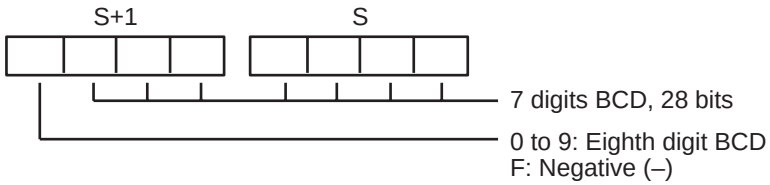
C = 0000 (Output Data Range: –999 9999 to 999 9999 BCD)



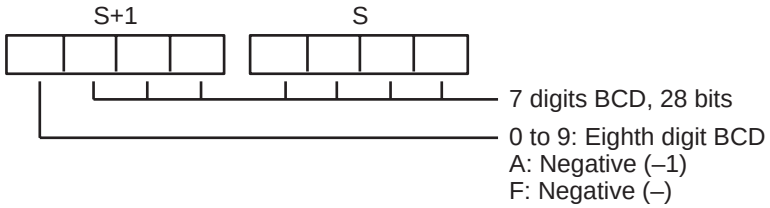
C = 0001 (Output Data Range: –7999 9999 to 7999 9999 BCD)



C = 0002 (Output Data Range: –999 9999 to 9999 9999 BCD)



C = 0003 (Output Data Range: –1999 9999 to 9999 9999 BCD)



The following table shows the possible double signed binary values for each signed BCD format. An error will occur if the source data is not within the allowed range for the specified signed BCD format.

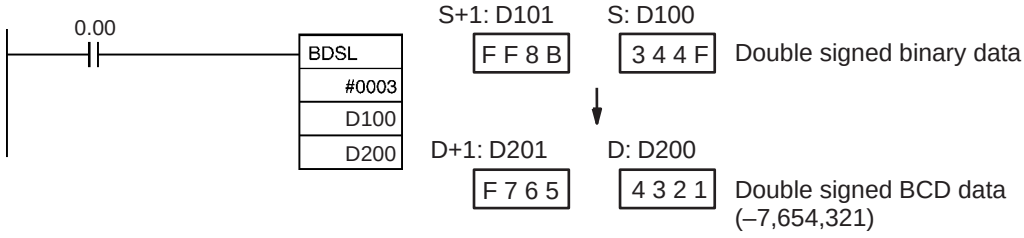
Setting	Signed binary values	Signed BCD values
C=0000	FF67 6981 to FFFF FFFF	–999 9999 to –1
	0000 0000 to 0098 967F	0 to 999 9999
C=0001	FB3B 4C01 to FFFF FFFF	–7999 9999 to –1
	0000 0000 to 04C4 B3FF	0 to 7999 9999
C=0002	FF67 6981 to FFFF FFFF	–999 9999 to –1
	0000 0000 to 05F5 E0FF	0 to 9999 9999
C=0003	FECE D301 to FFFF FFFF	–1999 9999 to –1
	0000 0000 to 05F5 E0FF	0 to 9999 9999

Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0000 and the source data is not within the range: FF67 6981 to FFFF FFFF or 0000 0000 to 0098 967F. ON if C=0001 and the source data is not within the range: FB3B 4C01 to FFFF FFFF or 0000 0000 to 04C4 B3FF. ON if C=0002 and the source data is not within the range: FF67 6981 to FFFF FFFF or 0000 0000 to 05F5 E0FF. ON if C=0003 and the source data is not within the range: FECE D301 to FFFF FFFF or 0000 0000 to 05F5 E0FF. OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if C=0000 or 0001 and the result's sign bit is ON after execution. ON if C=0002 and the leftmost digit of the result is F. ON if C=0003 and the leftmost digit of the result is A or F. OFF in all other cases.

Example

When CIO 0.00 is ON in the following example, the double signed binary data in D101 and D100 are checked against the format specified in the control word (0003). The source data is correct, so the double signed binary data in D101 and D100 is converted to double signed BCD and output to D201 and D200.

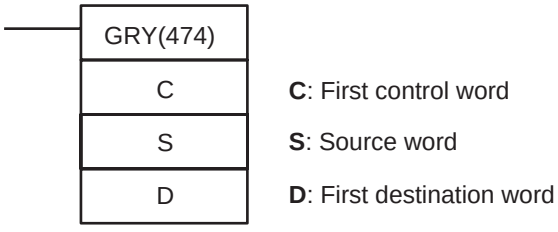


3-11-18 GRAY CODE CONVERT: GRY(474)

Purpose

Converts the gray binary code in a specified word to standard binary data, BCD data, or an angle at the specified resolution.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	GRY(474)
	Executed Once for Upward Differentiation	@GRY(474)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

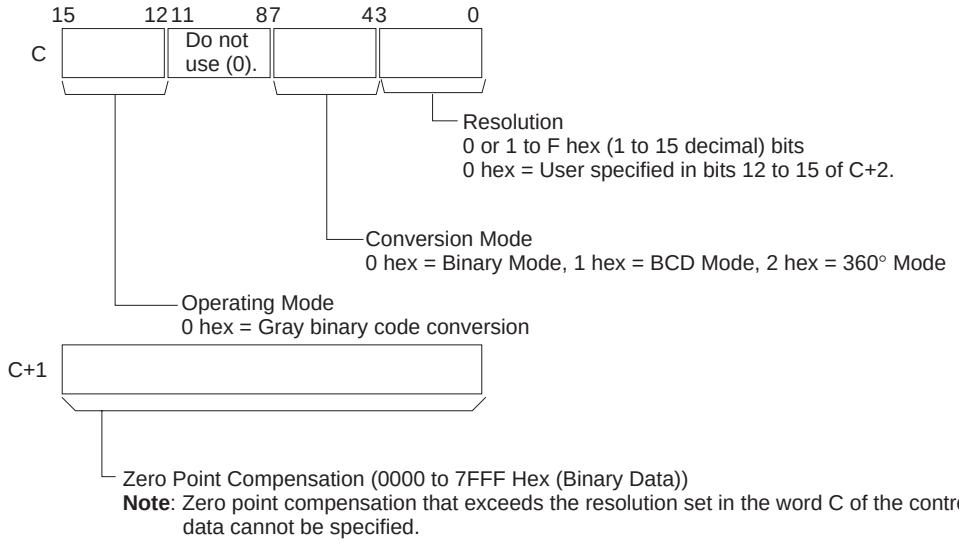
Applicable Program Areas

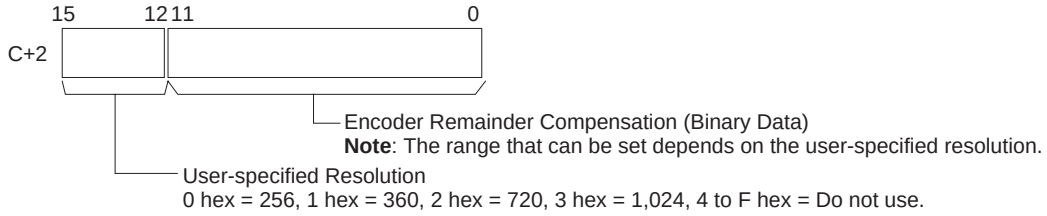
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

Specifies the parameters for the conversion as shown below.

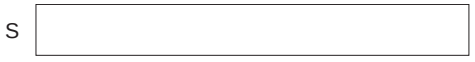




Note: The above setting is valid when the resolution is set to 0 hex in bits 00 to 03 of C.

S: Source Word

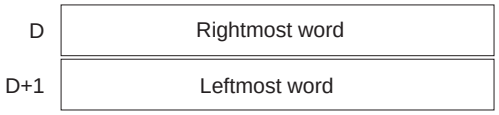
Contains the gray binary code to be converted. The range must be within the number of bits determined by the resolution specified in bits 00 to 03 of C. All bits outside of the number of bits for the specified resolution will be ignored. For example, if the specified resolution is 08 hex and S contains FFFF hex, the gray binary code will be taken as 00FF hex.



D: First destination word

Destination words D+1 and D contain the results of converting the gray binary code at the resolution specified in bits 00 to 03 of the control data word C and the conversion mode specified in bits 04 to 07 of the control data word C. The leftmost word is output to D+1 and the rightmost word is output to D. The ranges of data that are output are as follows:

Binary Mode: 0000 0000 to 0000 7FFF hex
BCD Mode:0000 0000 to 0003 2767
360° Mode:0000 0000 to 0000 3599
(0.0° to 359.9° in 0.1° increments, BCD)



Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143	CIO 0 to CIO 6142
Work Area	W0 to W510	W0 to W511	W0 to W510
Holding Bit Area	H0 to H510	H0 to H511	H0 to H510
Auxiliary Bit Area	A0 to A958	A0 to A959	A448 to A958
Timer Area	T0000 to T4094	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4094	C0000 to C4095	C0000 to C4094
DM Area	D0 to D32766	D0 to D32767	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

GRY(474) converts the gray binary code in the word specified in S at the resolution specified in C using one of the following conversion modes (binary, BCD, or 360°), also specified in C, and places the results in D and D+1.

Conversion mode	Function
Binary Mode	Gray binary code is converted to binary data between 0000 0000 and 0000 7FFF hex. Zero point offset and remainder compensation is applied and then the result is output to D and D+1.
BCD Mode	Gray binary code is converted to BCD data. Zero point offset and remainder compensation is applied, the data is converted to BCD between 0000 0000 and 0003 2767, and then the result is output to D and D+1.
360° Mode	Gray binary code is converted to BCD data. Zero point offset and remainder compensation is applied, the data is converted to an angle between 0000 0000 and 0000 3599 (0.0° to 359.9° in 0.1° increments), and then the result is output to D and D+1.

Note

(1) GRY(474) is normally used when inputting, through a DC Input Unit, a parallel signal (2^n) from an absolute encoder that outputs a gray binary code.

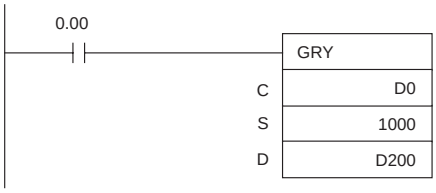
(2) If the word specified for S is allocated to an Input Unit, the input data converted by GRY(474) will be for the gray binary code from the previous CPU Unit cycle, i.e., it will be one cycle time old.

Flags

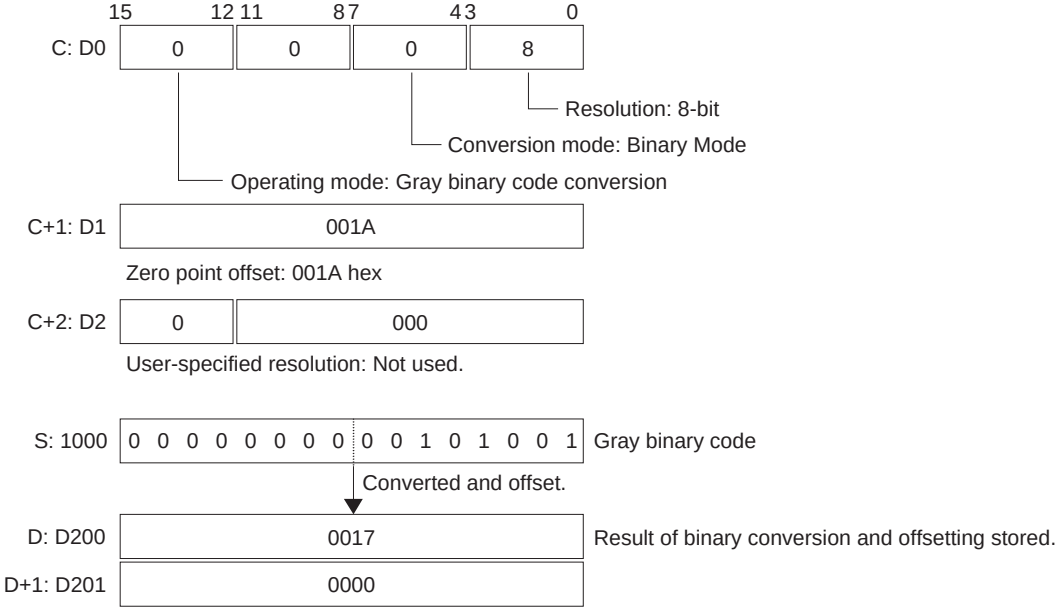
Name	Label	Operation
Error Flag	ER	ON if bits 12 to 15 of C are not 0 hex (operating mode = gray binary code conversion). ON if the zero point offset in C+1 is not within the specified resolution (including user-specified resolutions). ON if bits 04 to 07 of C are not 0 hex (= Binary Mode), 1 hex (= BCD Mode), or 2 hex (= 360° Mode). ON if the specified encoder remainder compensation exceeds the set user-specified resolution when bits 00 to 03 of C are 0 hex (= user-specified resolution). ON if the converted binary value is less than the encoder remainder compensation when bits 00 to 03 of C are 0 hex (= user-specified resolution). ON if the converted binary value is less than the resolution when bits 00 to 03 of C are 0 hex (= user-specified resolution). OFF in all other cases.
Equals Flag	=	OFF in all cases.
Negative Flag	N	OFF in all cases.

Examples

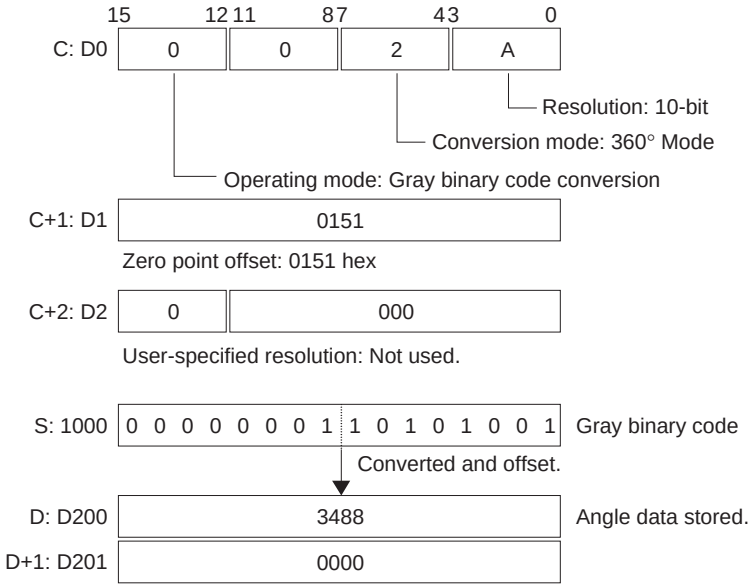
When CIO 0.00 is ON in the following example, the gray binary code in CIO 1000 is converted according to the settings in the control data in D0 to D2 and the result is output to D200.



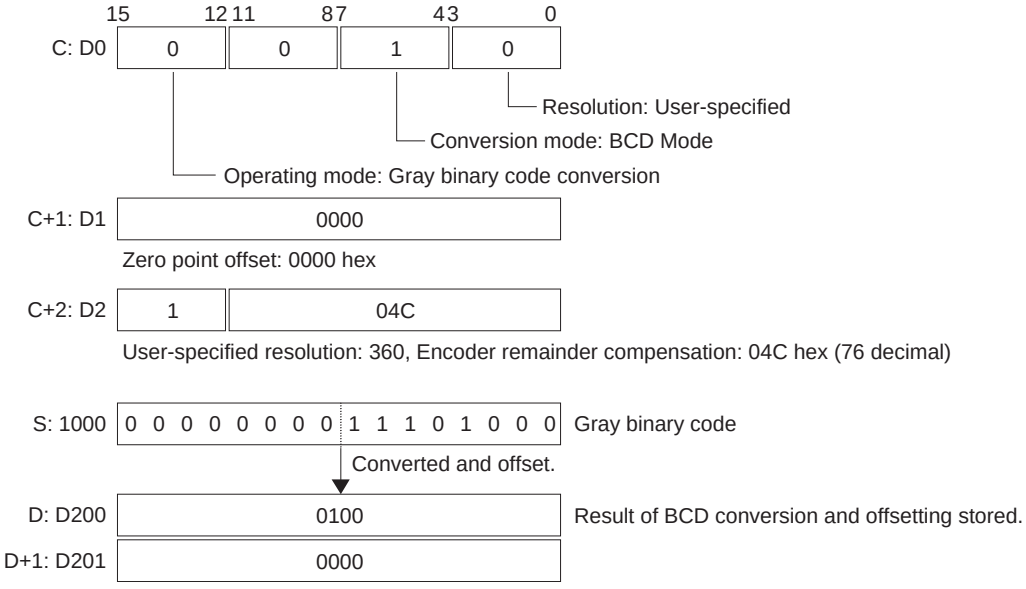
■ Example 1: Converting to Binary Data with an 8-bit Resolution and Zero Point Offset of 001A Hex



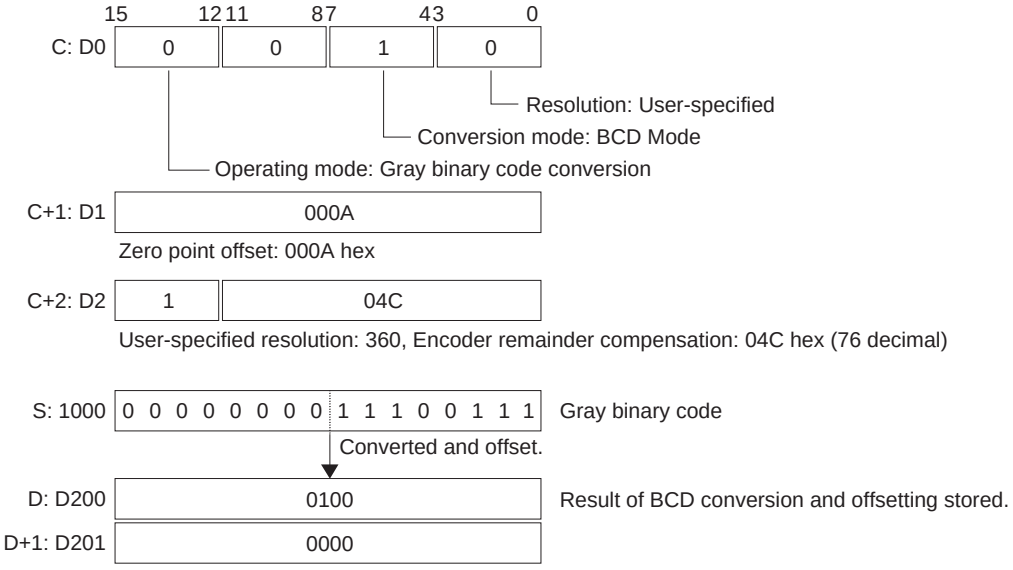
■ Example 2: Converting to Angle Data with a 10-bit Resolution and Zero Point Offset of 0151 Hex



■ Example 3: Converting to BCD Data with for an OMRON E6C2-AG5C Absolute Encoder (Resolution: 360/rotation, Encoder Remainder Compensation: 76) and Zero Point Offset of 0000 Hex



■ Example 4: Converting to BCD Data with for an OMRON E6C2-AG5C Absolute Encoder (Resolution: 360/rotation, Encoder Remainder Compensation: 76) and Zero Point Offset of 000A Hex



3-12 Logic Instructions

This section describes instructions which perform logic operations on word data.

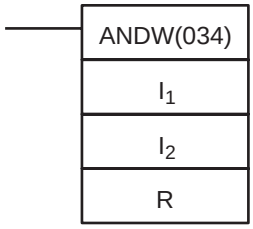
Instruction	Mnemonic	Function code	Page
LOGICAL AND	ANDW	034	437
DOUBLE LOGICAL AND	ANDL	610	438
LOGICAL OR	ORW	035	440
DOUBLE LOGICAL OR	ORWL	611	441
EXCLUSIVE OR	XORW	036	443
DOUBLE EXCLUSIVE OR	XORL	612	445
EXCLUSIVE NOR	XNRW	037	446
DOUBLE EXCLUSIVE NOR	XNRL	613	448
COMPLEMENT	COM	029	450
DOUBLE COMPLEMENT	COML	614	451

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.
Immediate Refreshing Specification		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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)		---

Area	I_1	I_2	R
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

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 content of corresponding bits in both I_1 and I_2 are 1 or when either 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 0. 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.

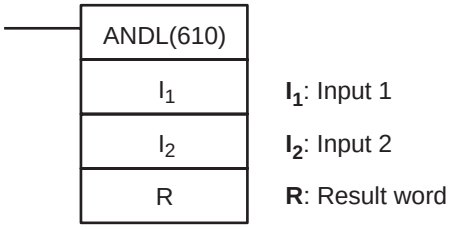
If as a result of the AND, the leftmost bit of R is 1, the Negative Flag will turn ON.

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



Variations

Variations	**Executed Each Cycle for ON Condition**	ANDL(610)
	Executed Once for Upward Differentiation	@ANDL(610)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

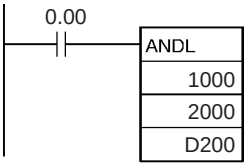
When ANDL(610) is executed, the Error Flag will turn OFF.

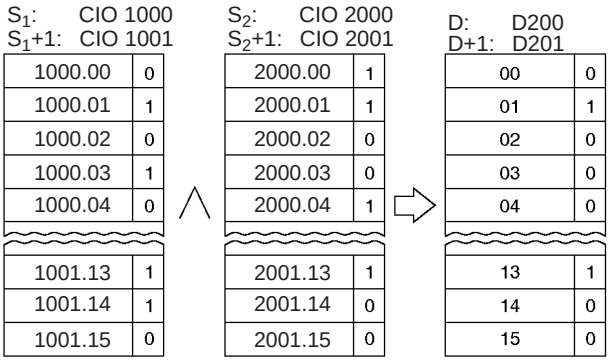
If as a result of the AND, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

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 0.00 is ON, the logical AND will be taken of corresponding bits in CIO 1001, CIO 1000 and CIO 2001, CIO 2000 and the results will be output to corresponding bits in D201 and D200.





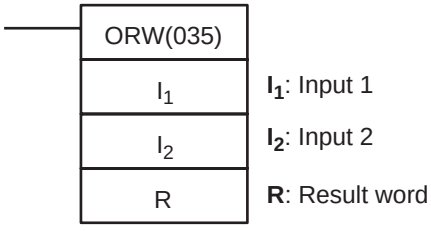
Note: The vertical arrow indicates logical AND.

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



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ORW(035)
	Executed Once for Upward Differentiation	@ORW(035)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		

Area	I_1	I_2	R
Indirect DM addresses in BCD	*D0 to *D32767		
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 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 are 1 or 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 0. 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.

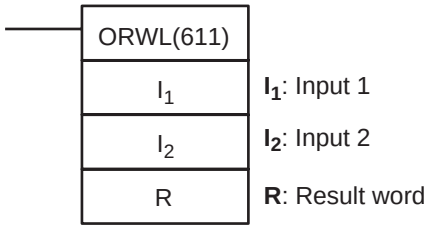
If as a result of the OR, the leftmost bit of R is 1, the Negative Flag will turn ON.

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.
Immediate Refreshing Specification		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 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ORWL(611) takes the logical OR of data specified in I_1 and I_2 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 +1are 1, a 1 will be output to the corresponding bit it R+1. When any of them are 0, a 0 will be output to the corresponding bit in 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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

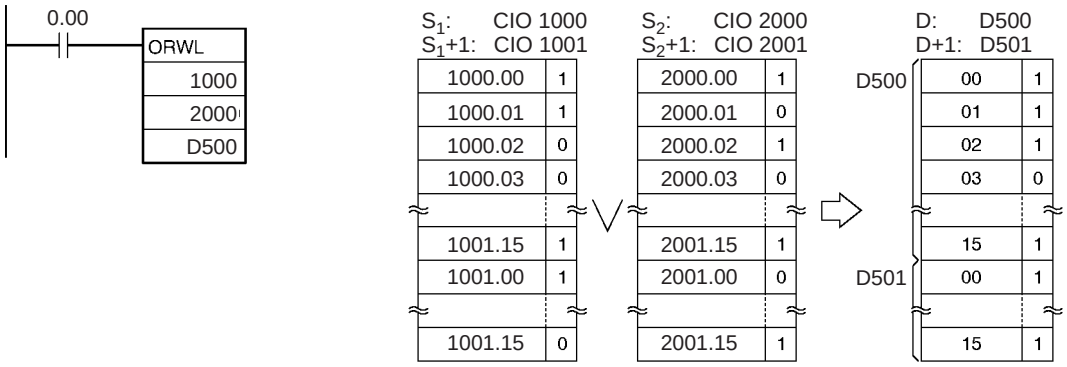
When ORWL(611) is executed, the Error Flag will turn OFF.

If as a result of the OR, the content of R, R+1 is 00000000 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 0.00 is ON, the logical OR will be taken of corresponding bits in CIO 1001, CIO 1000 and CIO 2001, CIO 2000 and the results will be output to corresponding bits in D501 and D500.



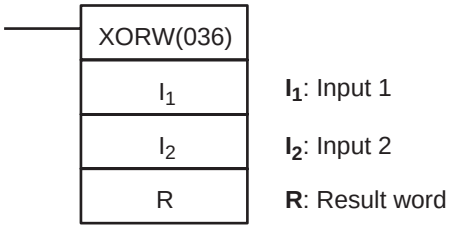
Note: The vertical arrow indicates logical OR.

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.
Immediate Refreshing Specification		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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		

Area	I_1	I_2	R
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
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 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 content of corresponding bits of I_1 and I_2 are different, a 1 will be output to the corresponding bit of R and when there are different, 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 0. 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.

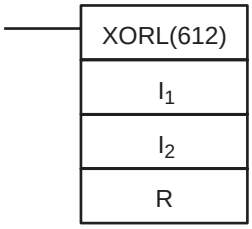
If as a result of the OR, the leftmost bit of R is 1, the Negative Flag will turn ON.

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



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	XORL(612)
	Executed Once for Upward Differentiation	@XORL(612)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

XORL(612) takes the logical exclusive OR of data specified in I_1 and I_2 as double-word data and outputs the result to R, R+1.

- When the content of any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1are different, a 1 will be output to the corresponding bit it R, R+1. When any 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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

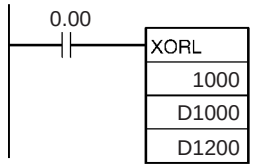
When XORL(612) is executed, the Error Flag will turn OFF.

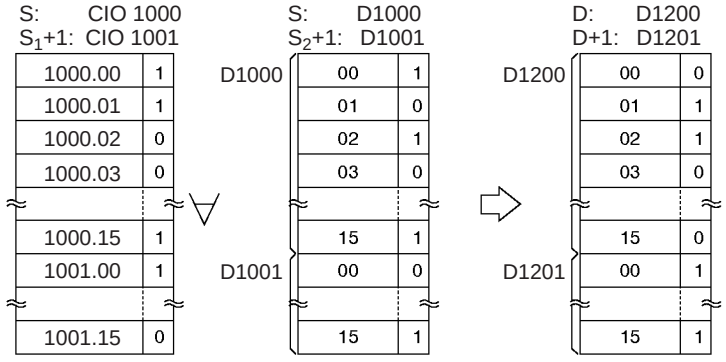
If as a result of the exclusive OR, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

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 0.00 is ON, the logical exclusive OR will be taken of corresponding bits in CIO 1001, CIO 1000 and D1001, D1000 and the results will be output to corresponding bits in D1201 and D1200.





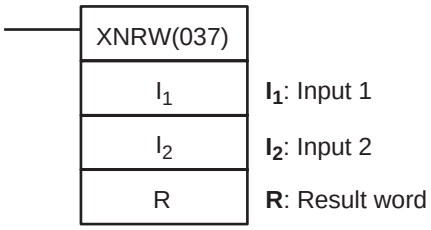
Note: The symbol indicates exclusive logical OR.

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



I_1: Input 1

I_2: Input 2

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	XNRW(037)
	Executed Once for Upward Differentiation	@XNRW(037)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
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 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 content 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 different, 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 0. 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.

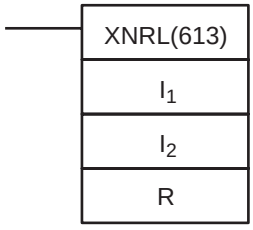
If as a result of the NOR, the leftmost bit of R is 1, the Negative Flag will turn ON.

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



I1: Input 1

I2: 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.
Immediate Refreshing Specification		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 0 to CIO 6142		
Work Area	W0 toW 510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		

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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

XNRL(613) takes the logical exclusive NOR of data specified in I_1 and I_2 and outputs the result to R, R+1.

- When the content of any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1are different, a 0 will be output to the corresponding bit in R, R+1. When any 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) + $\overline{(I_1, I_1+1)}$, $\overline{(I_2, I_2+1)}$ → (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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

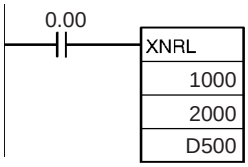
When XNRL(613) is executed, the Error Flag will turn OFF.

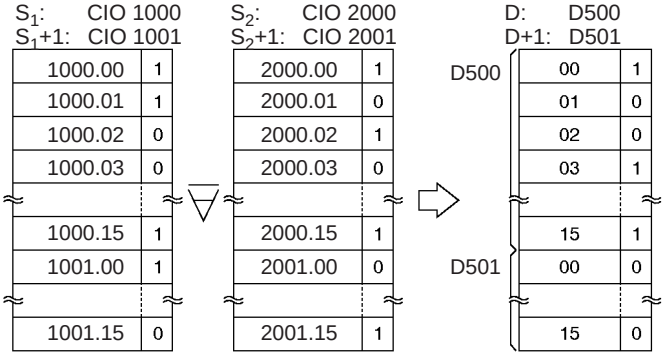
If as a result of the exclusive NOR, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of the exclusive NOR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0.00 is ON, the logical exclusive NOR will be taken of corresponding bits in CIO 1001, CIO 1000, and CIO 2001, CIO 2000 and the results will be output to corresponding bits in D501 and D500.





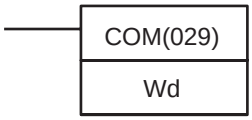
Note: The symbol indicates exclusive logical NOR.

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.
Immediate Refreshing Specification		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 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

COM(029) reverses the status of every specified bit in Wd.
$\overline{Wd} \rightarrow Wd$: $1 \rightarrow 0$ and $0 \rightarrow 1$

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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

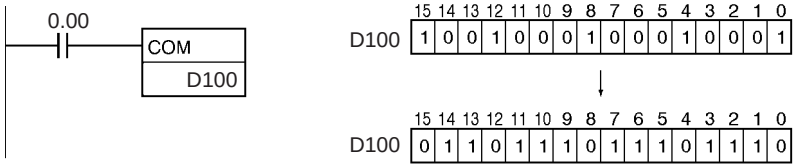
When COM(029) is executed, the Error Flag will turn OFF.

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 0.00 is ON in the following example, the status of each bit will be D100 is reversed.

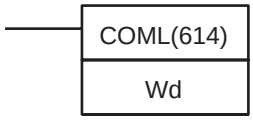


3-12-10 DOUBLE COMPLEMENT: COML(614)

Purpose

Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.

Ladder Symbol



Wd: 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.
Immediate Refreshing Specification		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 0 to CIO 6142
Work Area	W0 to W510
Holding Bit Area	H0 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

COML(614) reverses the status of every specified bit in Wd and Wd+1.
(Wd+1, Wd)→(Wd+1, Wd)

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 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

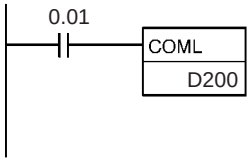
When COML(614) is executed, the Error Flag will turn OFF.

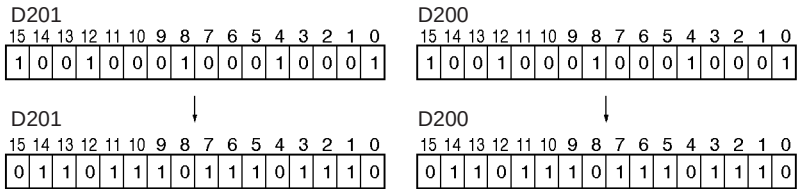
If as a result of COML, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of COML, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0.01 is ON in the following example, the status of each bit in D201 and D200 will be reversed.





3-13 Special Math Instructions

This section describes instructions used for special math calculations.

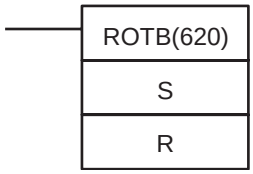
Instruction	Mnemonic	Function code	Page
BINARY ROOT	ROTB	620	452
BCD SQUARE ROOT	ROOT	072	454
ARITHMETIC PROCESS	APR	069	457
FLOATING POINT DIVIDE	FDIV	079	468
BIT COUNTER	BCNT	067	471

3-13-1 BINARY ROOT: ROTB(620)

Purpose

Computes the square root of the 32-bit signed binary contents (positive value) of the specified words and outputs the integer portion of the result to the specified result word.

Ladder Symbol



S: First source word

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	ROTB(620)
	Executed Once for Upward Differentiation	@ROTB(620)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

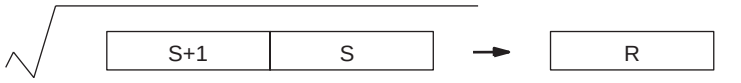
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W510	W0 to W511
Holding Bit Area	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A958	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

ROTB(620) computes the square root of the 32-bit binary number in S+1 and S and outputs the integer portion of the result to R. The non-integer remainder is eliminated.



Binary data (32 bits) Binary data (16 bits)

The range of data that can be specified for words S+1 and S is 0000 0000 to 3FFF FFFF. If a number from 4000 0000 to 7FFF FFFF is specified, it will be treated as 3FFF FFFF for the square root computation. An error will occur if the content of the source words is greater than 7FFF FFFF, i.e., if bit 15 of S+1 is 1.

Flags

Name	Label	Operation
Error Flag	ER	ON if bit 15 of S+1 is 1 (ON). OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

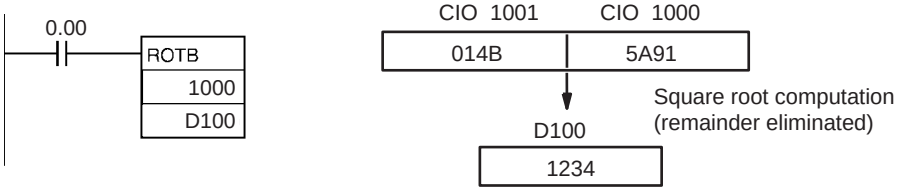
Name	Label	Operation
Overflow Flag	OF	ON if the content of S+1 and S is 4000 0000 to 7FFF FFFF. OFF in all other cases.
Underflow Flag	UF	OFF
Negative Flag	N	OFF

Precautions

The content of S+1 and S must be less than 8000 0000.
The operands of this instruction (S+1, S, and R) are all treated as binary values. If the input data is BCD, use the ROOT(072) instruction.

Example

When CIO 0.00 is ON in the following example, ROTB(620) calculates the square root of the data in CIO 1001 and CIO 1000, and writes the integer portion of the result in D100.

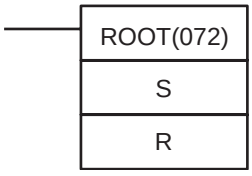


3-13-2 BCD SQUARE ROOT: ROOT(072)

Purpose

Computes the square root of an 8-digit BCD number and outputs the integer portion of the result to the specified result word.

Ladder Symbol



S: First source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ROOT(072)
	Executed Once for Upward Differentiation	@ROOT(072)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

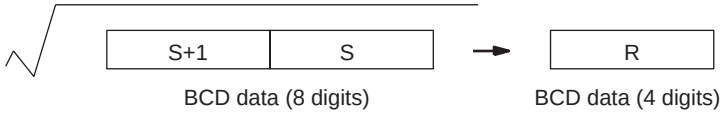
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W510	W0 to W511
Holding Bit Area	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A958	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767	

Area	S	R
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #99999999 (BCD)	---
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

ROOT(072) computes the square root of the 8-digit BCD number in S+1 and S and outputs the integer portion of the result to R. The non-integer remainder is eliminated.



Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

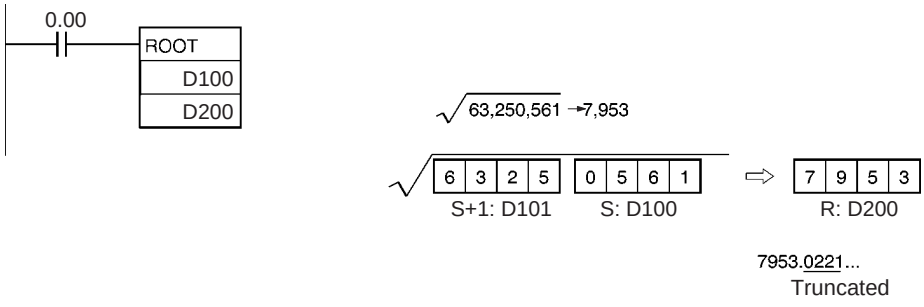
The operands of this instruction (S+1, S, and R) are all treated as BCD values. If the input data is binary, use the ROTB(620) instruction.

Examples

Square Root of 8-digit Number

When CIO 0.00 is ON in the following example, ROOT(072) calculates the square root of the data in D101 and D100, and writes the integer portion of the result in D200.

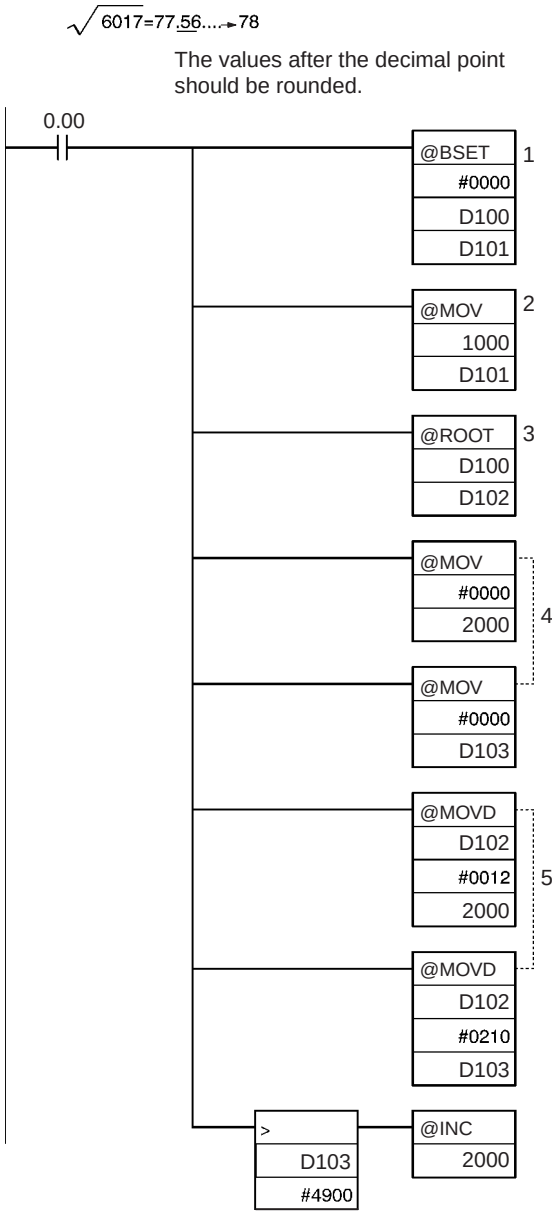
Note Figures after the decimal point are truncated for 8-digit numbers.



Square Root of a 4-digit Number

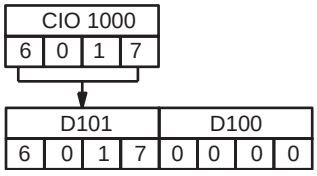
The following example shows how to take the square root of a 4-digit number and round off the result. This program example calculates the square root of the 4-digit number in CIO 1000, rounds off the result, and writes it to CIO 2000. (Basically, the 4-digit number is multiplied by 10,000 (100^2) and the result is divided by 100, increasing the precision of the calculation by a factor of 100.)

Note Figures after the decimal point are rounded for 4-digit numbers.

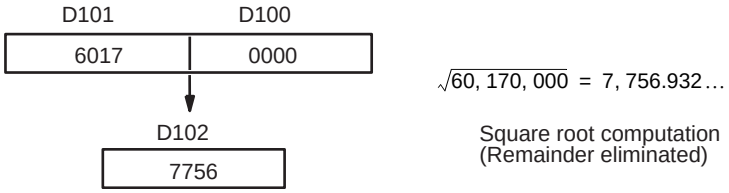


1,2,3... 1. The source words (D101 and D100) to be are cleared to 0000 0000.

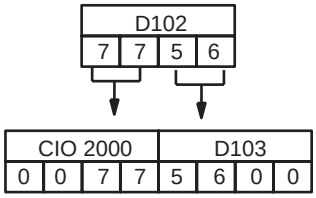
2. The 4-digit number is moved from CIO 1000 to D101.



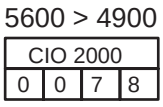
3. ROOT(072) calculates the square root of D101 and D100 and writes the result to D102.



4. D103 and the result word, CIO 2000, are cleared to 0000 0000.
5. The result of the square root calculation is divided by 100, with the integer portion written to CIO 2000 and the remainder going to D103.



6. If the content of D103 is greater than 4900, CIO 2000 is incremented by 1. In this case, the result is 78.

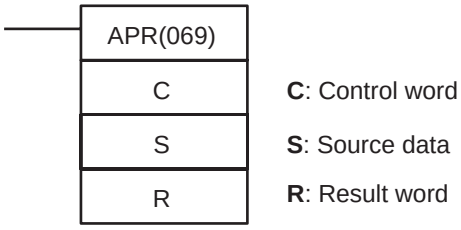


3-13-3 ARITHMETIC PROCESS: APR(069)

Purpose

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.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	APR(069)
	Executed Once for Upward Differentiation	@APR(069)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

Sine Function (C = 0000 Hex)

Operand	Value	Data range
C	0000 hex	---
S	0000 to 0900 (BCD)	0° to 90°
D	0000 to 9999 (BCD)	0.0000 to 0.9999
	9999 (BCD)	1.0000

Cosine Function (C = 0001 Hex)

Operand	Value	Data range
C	0001 hex	---
S	0000 to 0900 (BCD)	0° to 90°
D	0000 to 9999 (BCD)	0.0000 to 0.9999
	9999 (BCD)	1.0000

Linear Extrapolation Function (C = Data area address)

Operand	Value	Data range
C	Data area address	---
S	16-bit unsigned BCD data	0000 to 9999
	16-bit unsigned binary data	0 to 65,535
	16-bit signed binary data[1]	–32,768 to 32,767
	32-bit signed binary data[1]	–2,147,483,648 to 2,147,483,647
	Floating-point data[1]	$-\infty$, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, $+\infty$
D	16-bit unsigned BCD data	0000 to 9999
	16-bit unsigned binary data	0 to 65,535
	16-bit signed binary data[1]	–32,768 to 32,767
	32-bit signed binary data[1]	–2,147,483,648 to 2,147,483,647
	Floating-point data[1]	$-\infty$, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, $+\infty$

Note If C is a word address, APR(069) extrapolates the Y value for the X value in S based on coordinates (forming line segments) entered in advance in a table beginning at C. Refer to the *Description* section below for details.

Operand Specifications

Area	C	S	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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

The operation of APR(069) depends on the control word C. If C is 0000 or 0001, APR(069) computes the sine or cosine of S with S in units of tenths of degrees.

If C is a word address, APR(069) extrapolates the Y value for the X value in S based on coordinates (forming line segments) entered in advance in a table beginning at C.

Sine Function (C=0000)

When C is 0000, APR(069) calculates the SIN(S) and writes the result to R. The range for S is 0000 to 0900 BCD (0.0° to 90.0°) and the range for R is 0000 to 9999 BCD (0.0000 to 0.9999). The remainder of the result beyond the fourth decimal place is eliminated.

Cosine Function (C=0001)

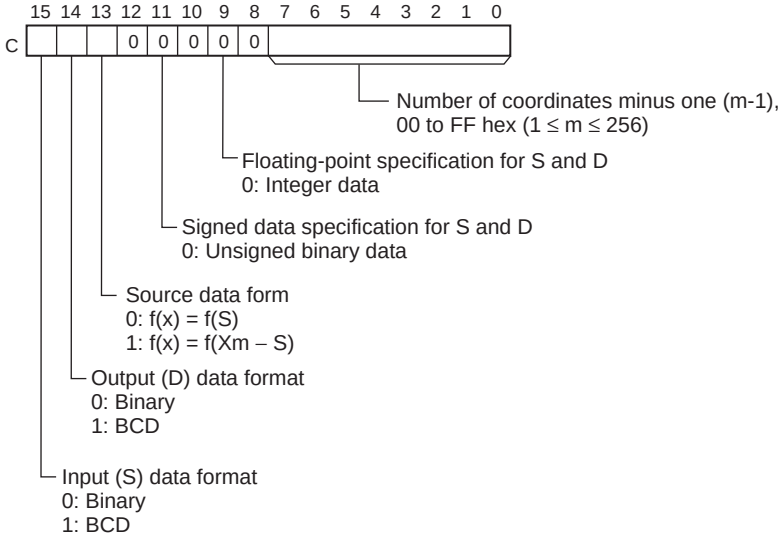
When C is 0001, APR(069) calculates the COS(S) and writes the result to R. The range for S is 0000 to 0900 BCD (0.0° to 90.0°) and the range for R is 0000 to 9999 BCD (0.0000 to 0.9999). The remainder of the result beyond the fourth decimal place is eliminated.

Linear Extrapolation

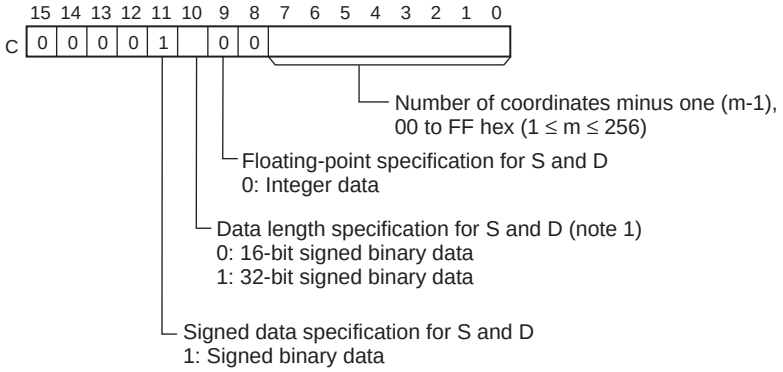
APR(069) linear extrapolation is specified when C is a word address.

The content of word C specifies the number of coordinates in a data table starting at C+2, the form of the source data, and whether data is BCD or binary.

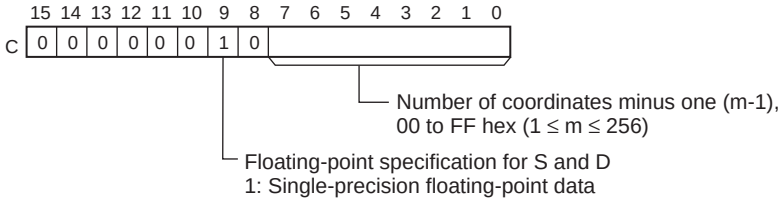
Unsigned Integer Data (Binary or BCD)



Signed Integer Data (Binary)

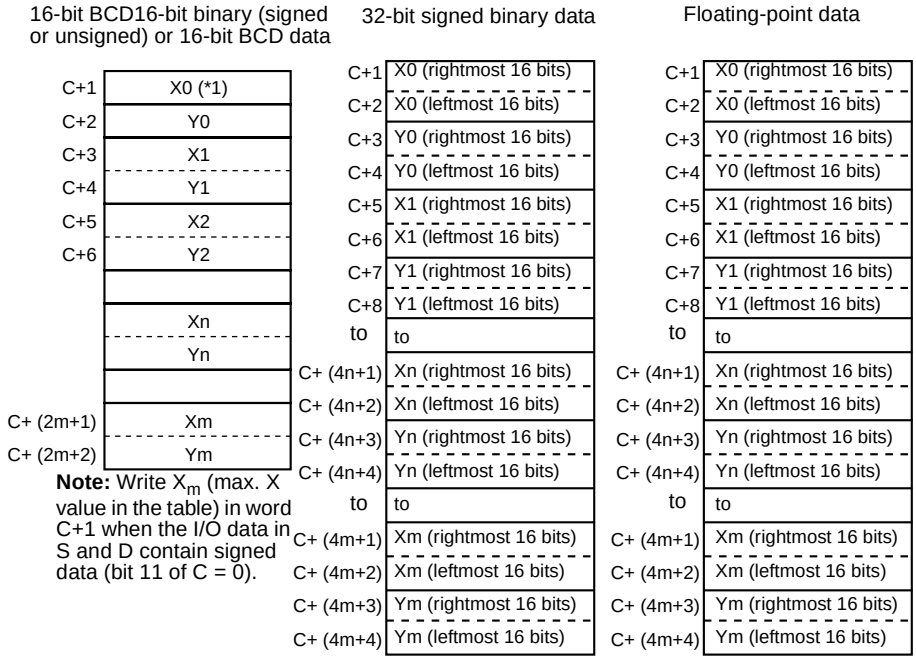


Single-precision Floating-point Data



If 16-bit binary or BCD data is being used, the line-segment data is contained in words C+1 through C+2m+2. If 32-bit binary or floating point data is being used, the line-segment data is contained in words C+1 through C+4m+4.

Bits 00 to 07 contain the number (binary) of line coordinates less 1, m–1. Bits 08 to 12 are not used. Bit 13 specifies either f(x)=f(S) or f(x)=f(X_m–S): OFF specifies f(x)=f(S) and ON specifies f(x)=f(X_m–S). Bit 14 determines whether the output is BCD or binary: OFF specifies binary and ON specifies BCD. Bit 15 determines whether the input is BCD or binary: OFF specifies binary and ON specifies BCD.

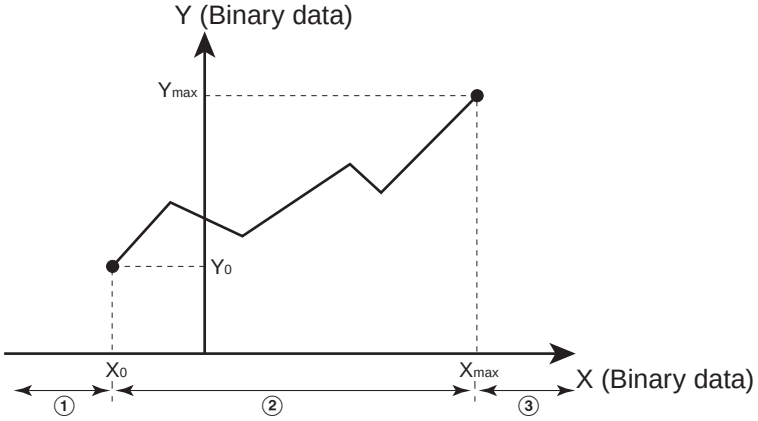


Note: Write X_m (max. X value in the table) in word C+1 when the I/O data in S and D contain signed data (bit 11 of C = 0).

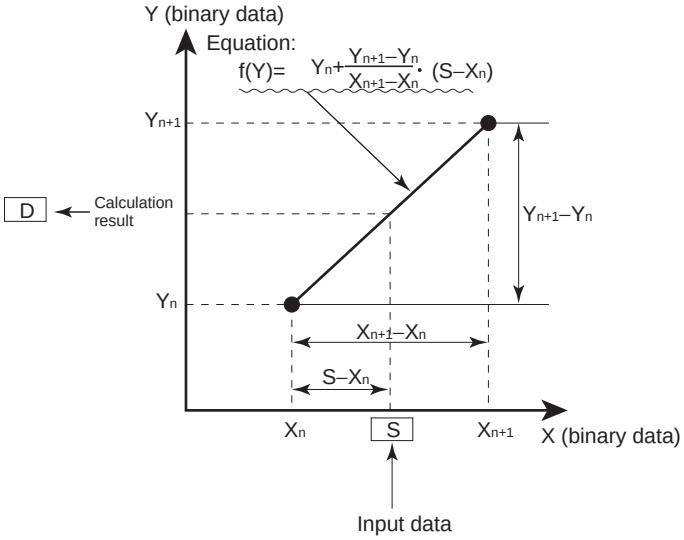
Note The X coordinates must be in ascending order: $X_1 < X_2 < ... < X_m$. Input all values of (X_n, Y_n) as binary data, regardless of the data format specified in control word C.

Operation 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 C+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 C+1. The following 5 kinds of I/O data can be used:

- 16-bit unsigned BCD data
- 16-bit unsigned binary data
- 16-bit signed binary data
- 32-bit signed binary data
- Single-precision floating-point data

Setting the Data Format in Control Word C

- 16-bit Unsigned BCD Data
  The input data and/or the output data can be 16-bit unsigned BCD data. Also, the linear extrapolation function can be set to operate on the value specified in S directly or on X_m–S. (X_m is the maximum value of X in the line-segment data.)

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary 1: BCD
Output data (D) format	14	0: Binary 1: BCD
Source data form	13	0: Operate on S 1: Operate on X_m–S
Signed data specification for S and D	11	0: Unsigned data
Data length specification for S and D	10	Invalid (fixed at 16 bits)
Floating-point specification	09	0: Integer data

- 16-bit Unsigned Binary Data
  The input data and/or the output data can be 16-bit unsigned binary data. Also, the linear extrapolation function can be set to operate on the value specified in S directly or on X_m–S. (X_m is the maximum value of X in the line-segment data.)

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary 1: BCD
Output data (D) format	14	0: Binary 1: BCD
Source data form	13	0: Operate on S 1: Operate on X_m–S
Signed data specification for S and D	11	0: Unsigned data
Data length specification for S and D	10	Invalid (fixed at 16 bits)
Floating-point specification	09	0: Integer data

- 16-bit Signed Binary Data

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary
Output data (D) format	14	0: Binary
Source data form	13	0
Signed data specification for S and D	11	1: Signed data
Data length specification for S and D	10	0: 16-bit signed binary data
Floating-point specification	09	0: Integer data

- 32-bit Signed Binary Data

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary
Output data (D) format	14	0: Binary
Source data form	13	0
Signed data specification for S and D	11	1: Signed data
Data length specification for S and D	10	1: 32-bit signed binary data
Floating-point specification	09	0: Integer data

Note If the "Data length specification for S and D" in bit 10 of C is set to 1 and a 16-bit constant is input for S, the input data will be converted to 32-bit signed binary before the linear extrapolation calculation.

- Floating-point Data

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary
Output data (D) format	14	0: Binary
Source data form	13	0
Signed data specification for S and D	11	0
Data length specification for S and D	10	0
Floating-point specification	09	1: Floating-point data

Note If the "Floating-point specification" in bit 09 of C is set to 1, a constant cannot be input for S.

Flags

Name	Label	Operation
Error Flag	ER	ON if C is a constant greater than 0001. ON if C is a word address but the X coordinates are not in ascending order ($X_1 \leq X_2 \leq ... \leq X_m$). ON if C is a word address and bits 9, 11, and 15 of C indicate BCD input, but S is not BCD. ON if C is a word address and bit 9 of C indicates floating-point data, but S is a one-word constant. ON if C is 0000 or 0001 but S is not BCD between 0000 and 0900. 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 R is ON. OFF in all other cases.

Precautions

The actual result for SIN(90°) and COS(0°) is 1, but 9999 (0.9999) will be output to R.

An error will occur if C is a constant greater than 0001.

An error will occur if linear extrapolation is specified but the X coordinates are not in ascending order ($X_1 < X_2 < ... < X_m$).

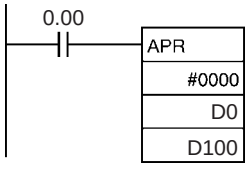
An error will occur if linear extrapolation is specified and BCD input is specified (bit 15 of C ON) but S is not BCD.

An error will occur if a trigonometric function is specified (C=0000 or 0001) but S is not BCD between 0000 and 0900.

Examples

Sine Function (C: #0000)

The following example shows APR(069) used to calculate the sine of 30°.



Source data

S: D0			
0	10^1	10^0	10^{-1}
0	3	0	0

Set the source data in 10^{-1} degrees. (0000 to 0900, BCD)

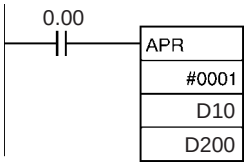
Result

R: D100			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
5	0	0	0

Result data has four significant digits, fifth and higher digits are ignored. (0000 to 9999, BCD)

Cosine Function (C: #0001)

The following example shows APR(069) used to calculate the cosine of 30°. (SIN(30) = 0.8660)



Source data

S: D10			
0	10^1	10^0	10^{-1}
0	3	0	0

Set the source data in 10^{-1} degrees. (0000 to 0900, BCD)

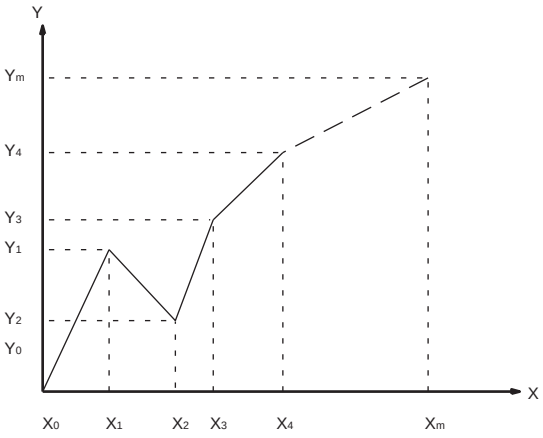
Result

R: D200			
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)

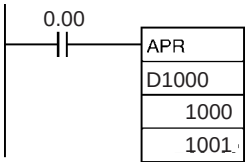
Linear Extrapolation (C: Word Address) Using 16-bit Unsigned BCD or Binary Data

APR(069) processes the input data specified in S based on the control data in C and the line-segment data specified in the table beginning at C+1. The result is output to D.

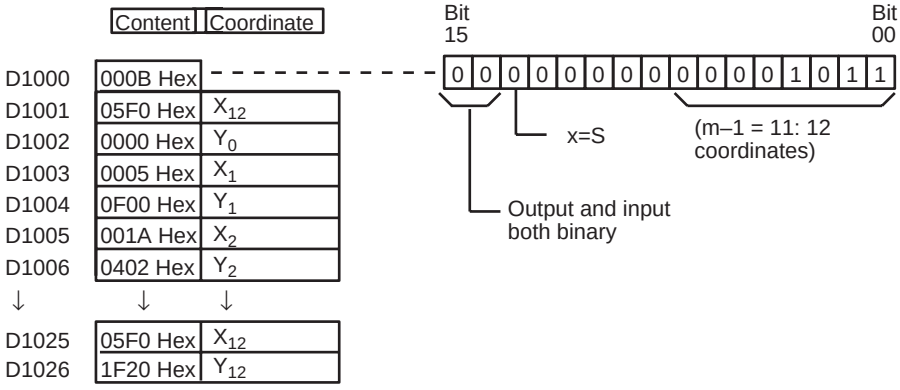


Word	Coordinate
C+1	Xm (max. X value)
C+2	Y_0
C+3	X_1
C+4	Y_1
C+5	X_2
C+6	Y_2
↓	↓
C+(2m+1)	X_m (max. X value)
C+(2m+2)	Y_m

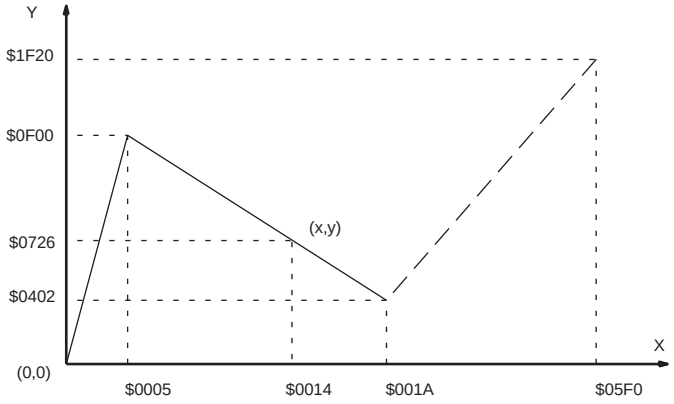
- $Y_n = f(X_n)$, $Y_0 = f(X_0)$
- Be sure that $X_{n-1} < X_n$ in all cases.
- Input all values of (X_n, Y_n) as binary data.



This example shows how to construct a linear extrapolation with 12 coordinates. The block of data is continuous, as it must be, from D1000 to D1026 (C to C + (2 × 12 + 2)). The input data is taken from CIO 1000, and the result is output to CIO 1001.



In this case, the source word, CIO 1000, contains 0014, and f(0014) = 0726 is output to R, CIO 1001.



The linear-extrapolation calculation is shown below.

$$Y = 0F00 + \frac{0402 - 0F00}{001A - 0005} \times (0014 - 0015)$$

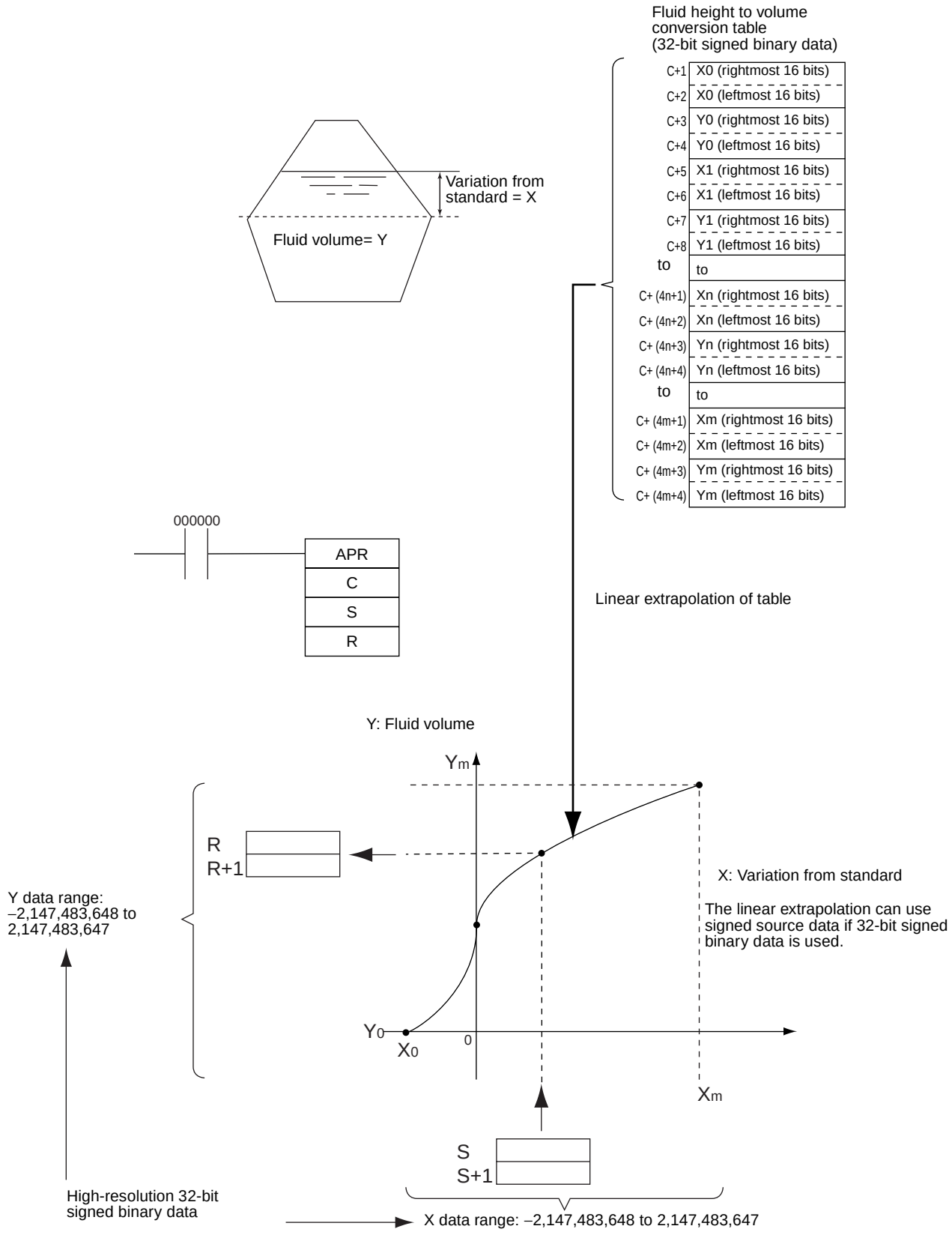
$$= 0F00 - (0086 \times 000F)$$

$$= 0726$$

Values are all hexadecimal (hex).

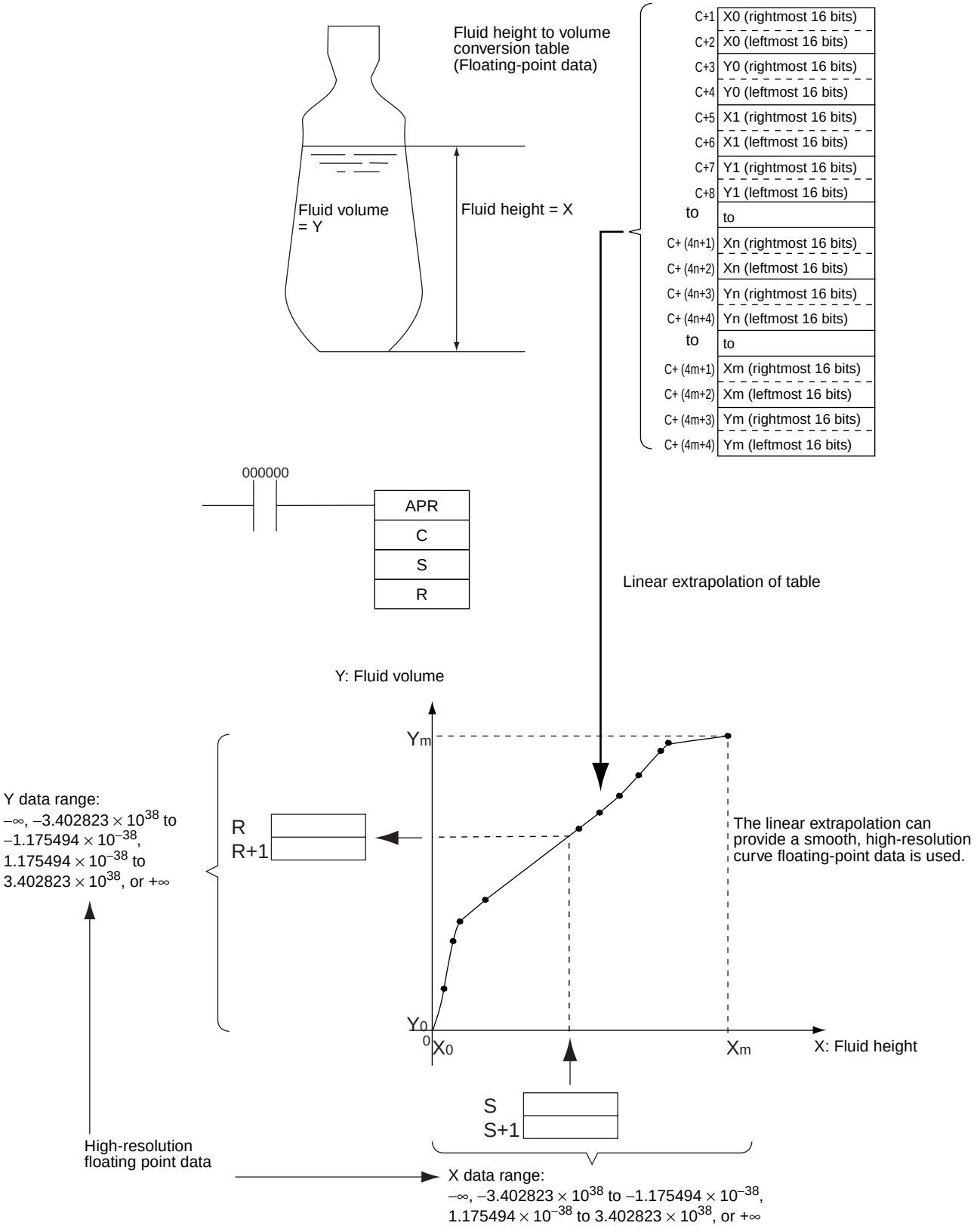
Linear Extrapolation (C: Word Address) Using 32-bit Signed Binary Data

In this example, APR(069) is used to convert the fluid height in a tank to fluid volume based on the shape of the holding tank.



Linear Extrapolation (C: Word Address) Using Floating-point Data

In this example, APR(069) is used to convert the fluid height in a tank to fluid volume based on the shape of the holding tank.

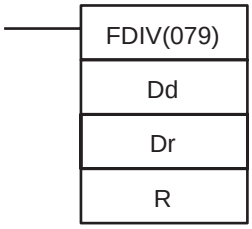


3-13-4 FLOATING POINT DIVIDE: FDIV(079)

Purpose

Divides one 7-digit floating-point number by another. The floating-point numbers are expressed in scientific notation (7-digit mantissa and 1-digit exponent).

Ladder Symbol



Dd: First dividend word

Dr: First divisor word

R: First result word

Variations

Variations	Executed Each Cycle for ON Condition	FDIV(079)
	Executed Once for Upward Differentiation	@FDIV(079)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

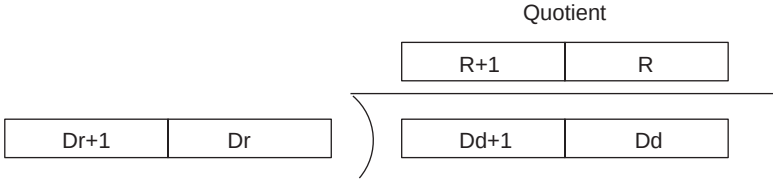
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

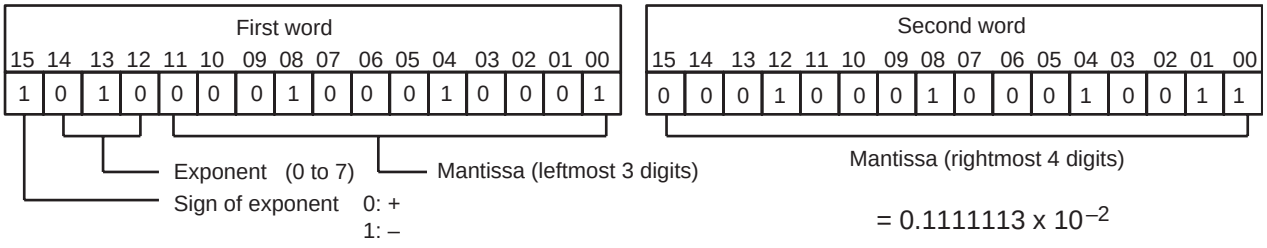
Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

FDIV(079) divides the floating-point value in Dd and Dd+1 by that in Dr and Dr+1 and places the result in R and R+1.



To represent the floating-point values, the rightmost seven digits are used for the mantissa and the leftmost digit is used for the exponent, as shown in the diagram below. The leftmost digit can range from 0 to F; positive exponents range from 0 to 7 and negative exponents range from 8 to F (0 to –7). The rightmost 7 digits must be BCD.



Two more examples of floating-point values are:

6123 4567: 0.1234567×10^{6} (6 = 0110 binary)

B123 4567: 0.1234567×10^{-3} (B = 1011 binary)

The following table shows the maximum and minimum values allowed.

Limit	8-digit hexadecimal	Floating-point
Maximum value	7999 9999	0.9999999×10^{7}
Minimum value (Divisor and dividend)	F000 0001	0.0000001×10^{-7}
Minimum value (Result)	F100 0000	0.1000000×10^{-7}

Flags

Name	Label	Operation
Error Flag	ER	ON if the mantissa (leftmost 7 digits) in Dd+1 and Dd is not BCD. ON if the mantissa (leftmost 7 digits) in Dr+1 and Dr is not BCD. ON if the divisor (Dr+1 and Dr) is 0. ON if the result is not between 0.1000000×10^{-7} and 0.9999999×10^{7}. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.

Precautions

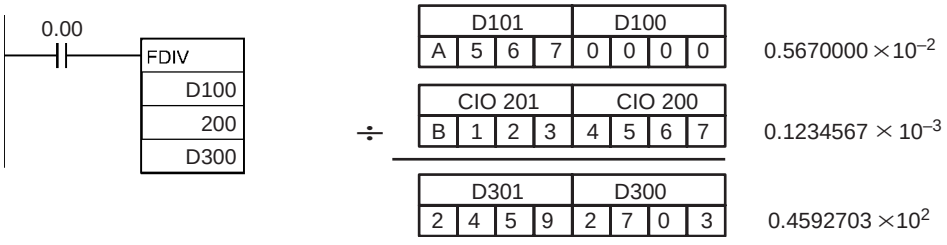
The result is expressed as a floating-point value, so it has 7 significant digits. The eighth and higher digits are eliminated.

The result must be between 0.1000000×10^{-7} and 0.9999999×10^{7}.

Examples

Basic Floating-point Division

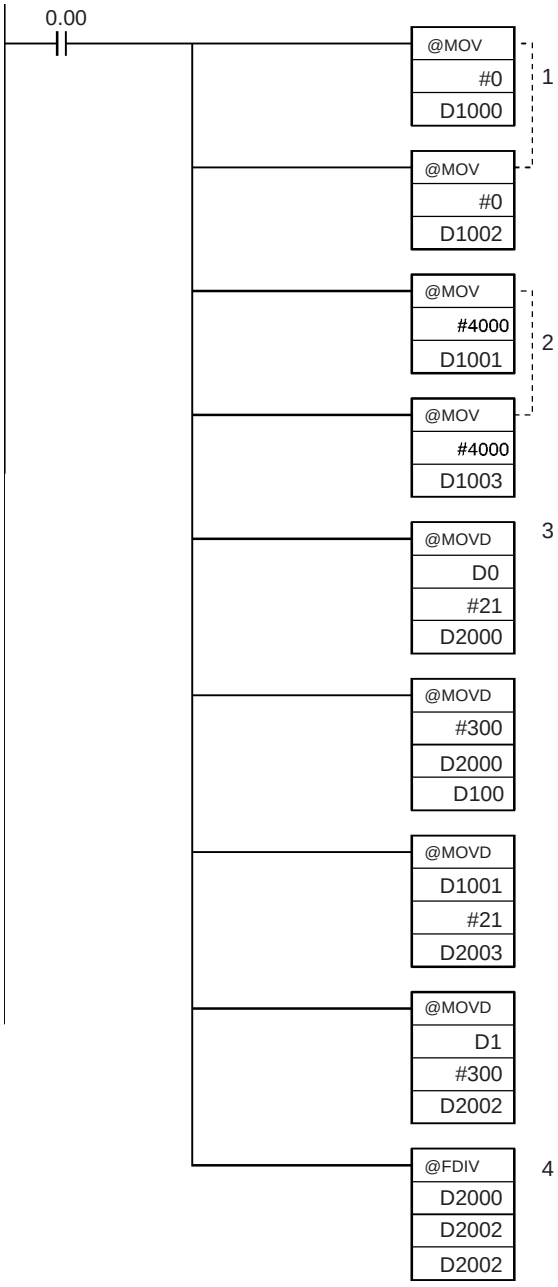
When CIO 0.00 is ON in the following example, FDIV(079) divides the floating-point number in D101 and D100 by the floating-point number in CIO 201 and CIO 200 and writes the result to D301 and D300.



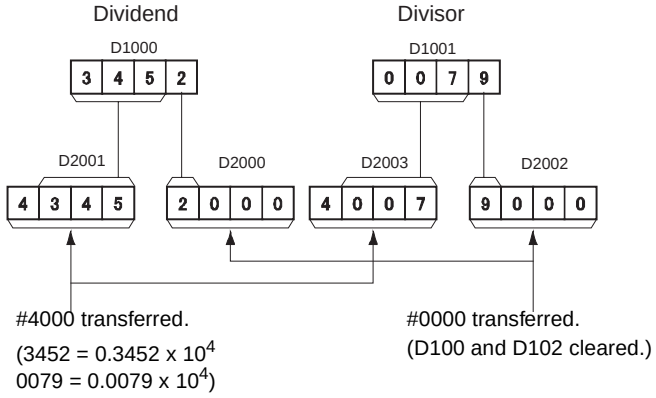
Floating-point Division of Two BCD Numbers

In this example, the 4-digit BCD number in D0 is divided by the 4-digit BCD number in D1 and the floating-point result is written to D2 and D3.

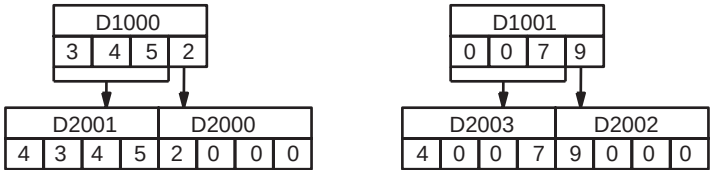
To perform the floating point division, the BCD value in D0 is converted to floating-point format in D101 and D100 and the BCD value in D1 is converted to floating-point format in D103 and D102.



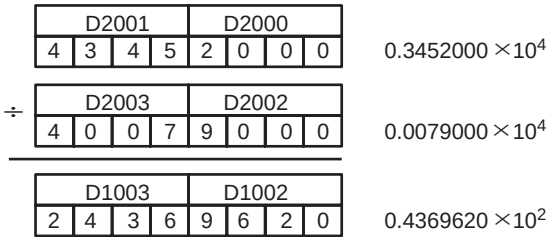
1,2,3... 1. D100 and D1002 are set to 0000 and D1001 and D1003 are set to 4000.



2. MOVD(083) is used to move the digits of the original source words to the proper digits in the 2-word floating-point formats.



3. FDIV(079) divides the floating-point number in D101 and D100 by the floating-point number in D103 and D102.

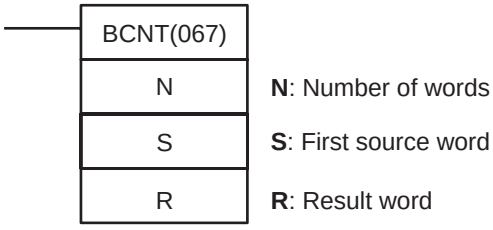


3-13-5 BIT COUNTER: BCNT(067)

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.
Immediate Refreshing Specification		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

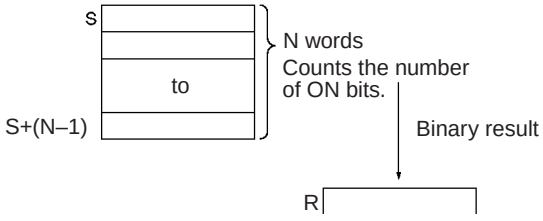
S and S+(N–1) must be in the same data area.

Operand Specifications

Area	N	S	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0001 to #FFFF (binary) or &1 to &65,535	---	
Data Registers	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 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

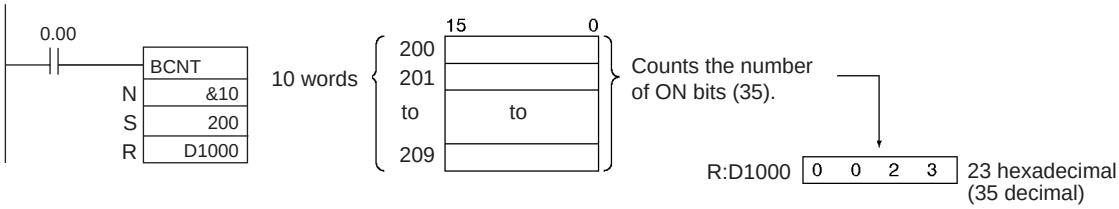
Name	Label	Operation
Error Flag	ER	ON if N is 0000. ON if result exceeds FFFF. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

An error will occur if N=0000 or the result exceeds FFFF.

Example

When CIO 0.00 is ON in the following example, BCNT(067) counts the total number of ON bits in the 10 words from CIO 200 and CIO 209 and writes the result to D1000.



3-14 Floating-point Math Instructions

The Floating-point Math Instructions convert data and perform floating-point arithmetic operations.

Instruction	Mnemonic	Function code	Page
FLOATING TO 16-BIT	FIX	450	479
FLOATING TO 32-BIT	FIXL	451	481
16-BIT TO FLOATING	FLT	452	482
32-BIT TO FLOATING	FLTL	453	484
FLOATING-POINT ADD	+F	454	485
FLOATING-POINT SUB-TRACT	–F	455	487
FLOATING-POINT MULTI-PLY	*F	456	489
FLOATING-POINT DIVIDE	/F	457	491
DEGREES TO RADIANS	RAD	458	493
RADIANS-TO-DEGREES	DEG	459	494
SINE	SIN	460	496
COSINE	COS	461	497
TANGENT	TAN	462	499
ARC SINE	ASIN	463	501
ARC COSINE	ACOS	464	503
ARC TANGENT	ATAN	465	504
SQUARE ROOT	SQRT	466	506
EXPONENT	EXP	467	508
LOGARITHM	LOG	468	510
EXPONENTIAL POWER	PWR	840	512

Refer to *3-15-21 Double-precision Floating-point Input Instructions* for details on double-precision floating-point instructions.

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	513
FLOATING-POINT TO ASCII	FSTR	448	517
ASCII TO FLOATING-POINT	FVAL	449	522

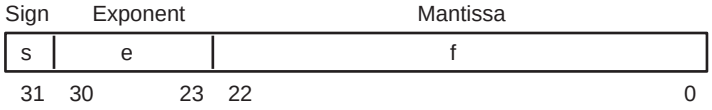
Data Format

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.

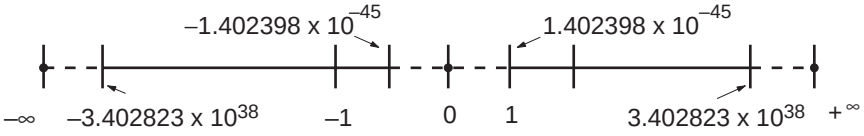
Number of Digits

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:

- $-\infty$
- $-3.402823 \times 10^{38} \leq \text{value} \leq -1.402398 \times 10^{-45}$
- 0
- $1.402398 \times 10^{-45} \leq \text{value} \leq 3.402823 \times 10^{38}$
- $+\infty$
- Not a number (NaN)



Special Numbers

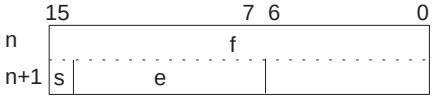
The formats for NaN, ±∞, and 0 are as follows:

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.

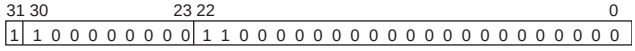
The exponent (e) will be expressed from 1 to 254, and the real exponent will be 127 less, i.e., –126 to 127.

The mantissa (f) will be expressed from 0 to $2^{33} – 1$, and it is assume that, in the real mantissa, bit 2^{33} 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.

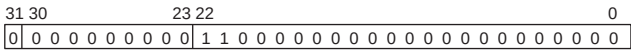
The exponent (e) will be 0, and the real exponent will be –126.

The mantissa (f) will be expressed from 1 to $2^{33} – 1$, and it is assume that, in the real mantissa, bit 2^{33} 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: -0.75×2^{-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 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.

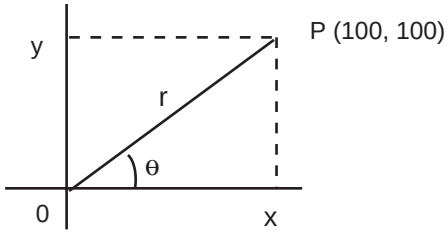
Floating-point Calculation Results

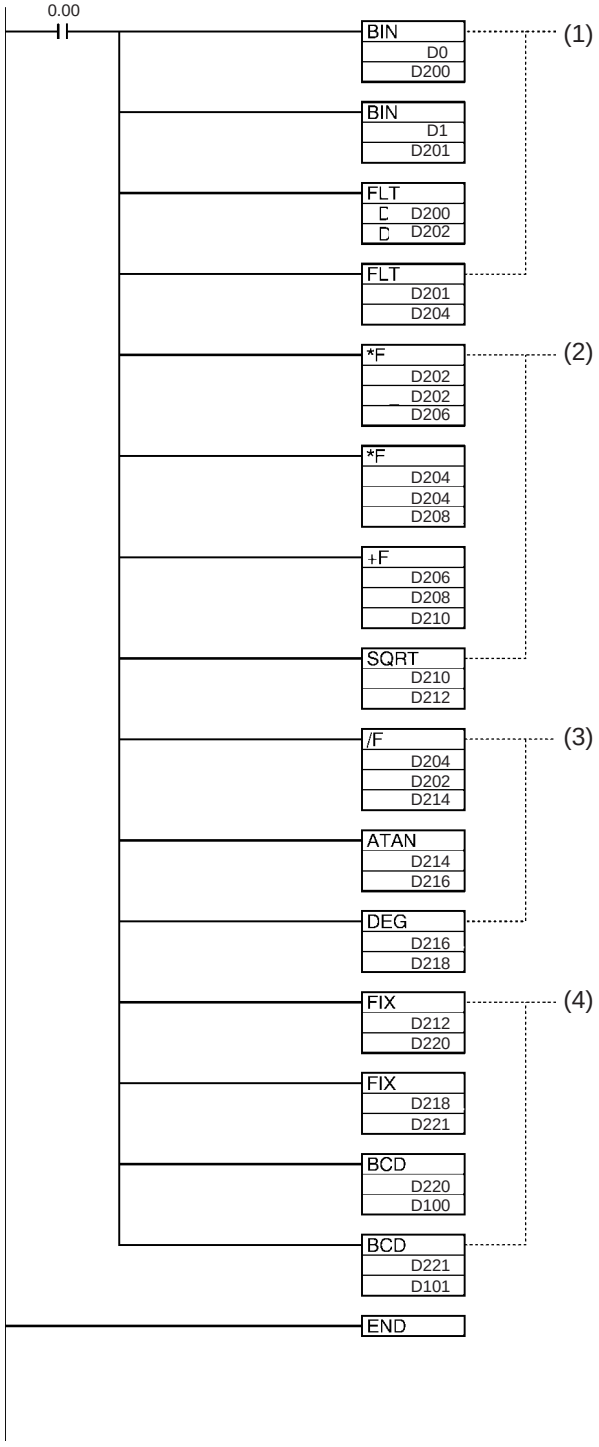
When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞. If the result is positive, it will be output as +∞; if negative, then –∞.

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 4-digit BCD content of D0 and D1. The distance (r) from the origin and the angle (θ, in degrees) are found and output to D100 and D101. In the result, everything to the right of the decimal point is truncated.




0.00
BIN
D0
D200
(1)
BIN
D1
D201
FLT
D200
D202
FLT
D201
D204
*F
D202
D202
D206
(2)
*F
D204
D204
D208
+F
D206
D208
D210
SQRT
D210
D212
/F
D204
D202
D214
(3)
ATAN
D214
D216
DEG
D216
D218
FIX
D212
D220
(4)
FIX
D218
D221
BCD
D220
D100
BCD
D221
D101
END

Calculations

Distance r = $\sqrt{x^2 + y^2}$

Angle = $\tan^{-1}\left(\frac{y}{x}\right)$

Example

Distance r = $\sqrt{100^2 + 100^2} = 141.4214$

Angle = $\tan^{-1}\left(\frac{100}{100}\right) \times \left(\frac{180}{\pi}\right) = 45.0$

DM Contents

D0	#0100 (BCD)	x	→	D100	0 1 4 1 (BCD) r
D1	#0100 (BCD)	y		D101	0 0 4 5 (BCD)

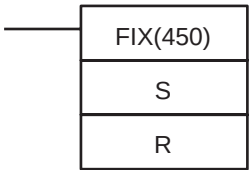
1. This section of the program converts the data from BCD to floating-point.
   a. The data area from D200 onwards is used as a work area.
   b. First BIN(023) is used to temporarily convert the BCD data to binary data, and then FLT(452) is used to convert the binary data to floating-point data.
   c. The value of x that has been converted to floating-point data is output to D203 and D202.
   d. The value of y that has been converted to floating-point data is output to D205 and D204.
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 D213 and D212 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 D219 and D218 as floating-point data.
4. The data is converted back from floating-point to BCD.
   a. First FIX(450) is used to temporarily convert the floating-point data to binary data, and then BCD(024) is used to convert the binary data to BCD data.
   b. The distance r is output to D100.
   c. The angle θ is output to D101.

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



S: First source word

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	FIX(450)
	Executed Once for Upward Differentiation	@FIX(450)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W510	W0 to W511
Holding Bit Area	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A958	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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 R.

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

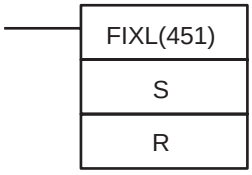
The content of S+1 and S must be floating-point data and the integer portion must be in the range of –32,768 to 32,767.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

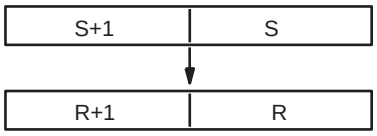
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.



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.
A floating-point value of –214,748,340.5 is converted to –214,748,340.

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

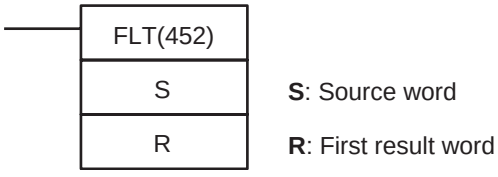
The content of S+1 and S must be floating-point data and the integer portion must be in the range of –2,147,483,648 to 2,147,483,647.

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



Variations

Variations		
	Executed Each Cycle for ON Condition	FLT(452)
	Executed Once for Upward Differentiation	@FLT(452)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

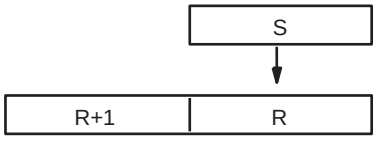
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142
Work Area	W0 to W511	W0 to W510

Area	S	R
Holding Bit Area	H0 to H511	H0 to H510
Auxiliary Bit Area	A0 to A959	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4095	C0000 to C4094
DM Area	D0 to D32767	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
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 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.



Signed binary data (16 bits)

Floating-point data (32 bits)

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

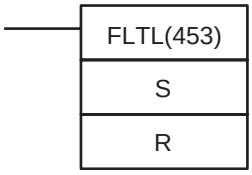
The content of S must contain signed binary data with a (decimal) value in the range of –32,768 to 32,767.

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



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FLTL(453)
	Executed Once for Upward Differentiation	@FLTL(453)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

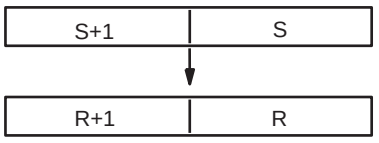
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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 (32 bits)

Floating-point data (32 bits)

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 –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

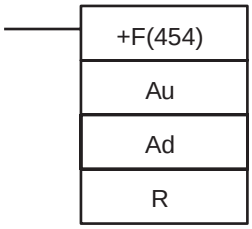
The result will not be exact if a number with an absolute value greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted.

3-14-5 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

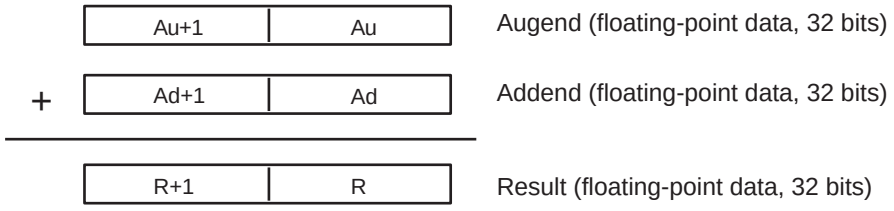
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		

Area	Au	Ad	R
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.

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

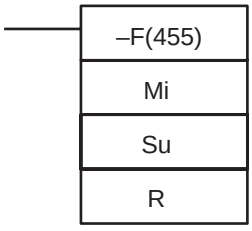
The augend (Au+1 and Au) and Addend (Ad+1 and Ad) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

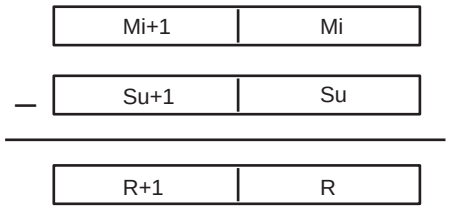
Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		

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 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

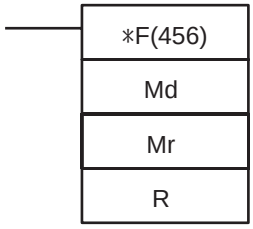
The Minuend (Mi+1 and Mi) and Subtrahend (Su+1 and Su) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

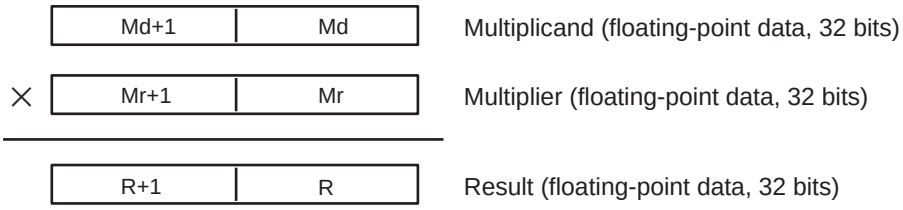
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.	See note 2.
Numeral	0	See note 1.	+/–∞	+/–∞	
+∞	See note 2.	+/–∞	+∞	–∞	
–∞	See note 2	+/–∞	–∞	+∞	
NaN					

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

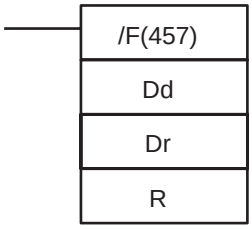
The Multiplicand (Md+1 and Md) and Multiplier (Mr+1 and Mr) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

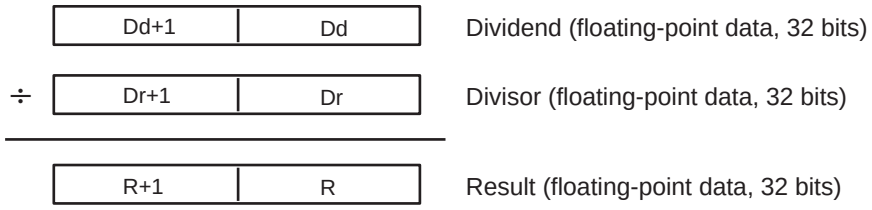
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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

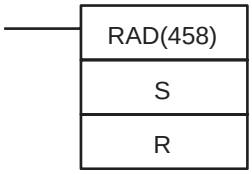
The Dividend (Dd+1 and Dd) and Divisor (Dr+1 and Dr) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

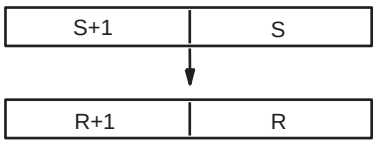
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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 × π/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

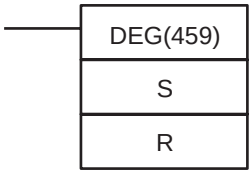
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

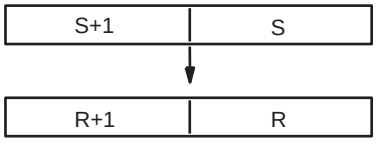
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	

Area	S	R
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000000 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 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

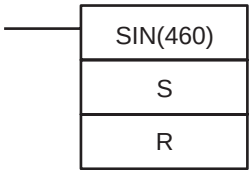
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

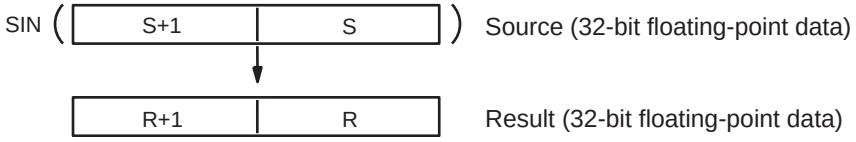
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

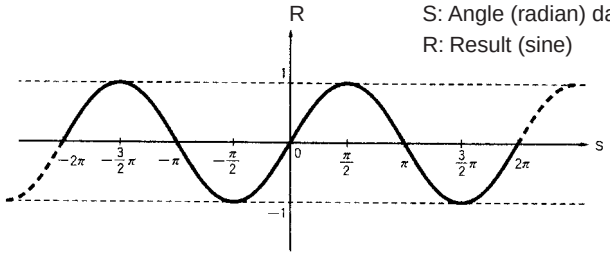
Description

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 LOGARITHM: LOG(468) 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

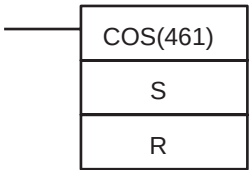
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

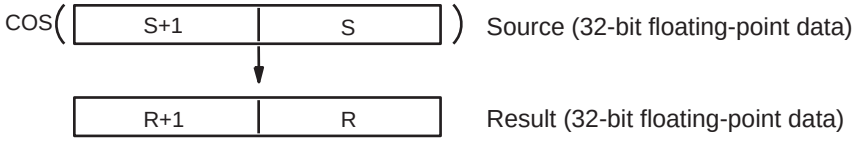
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

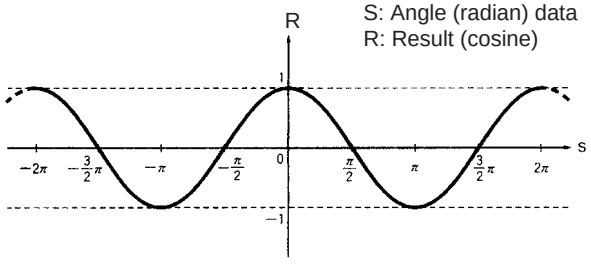
Description

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 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

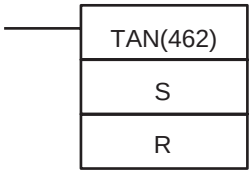
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-13 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

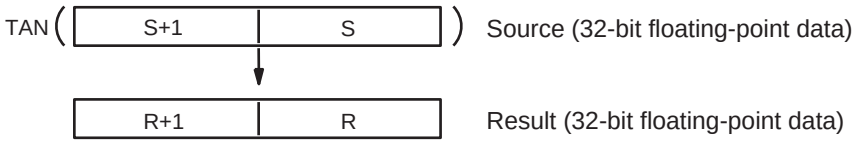
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	

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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

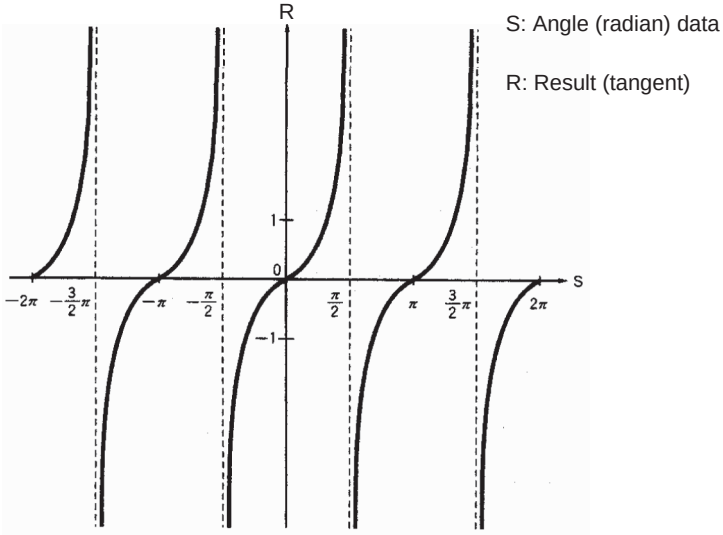
TAN(462) calculates the tangent of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



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.



S: Angle (radian) data

R: Result (tangent)

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

Name	Label	Operation
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

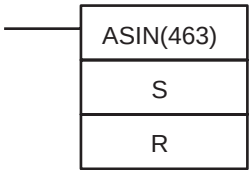
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

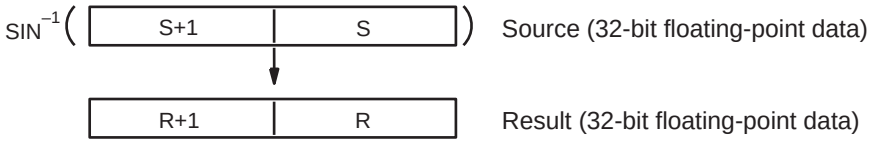
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

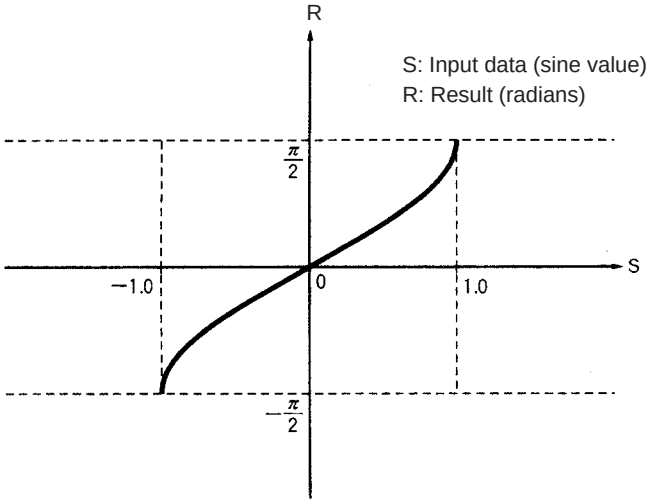
ASIN(463) computes the angle (in radians) for a sine value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



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

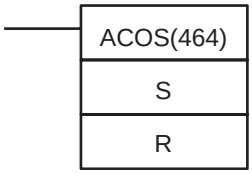
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

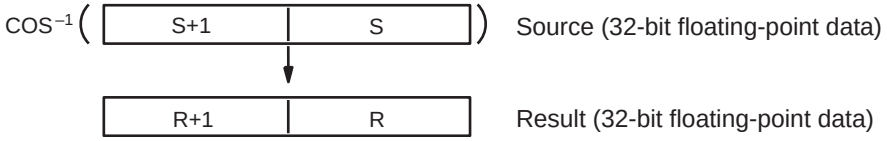
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

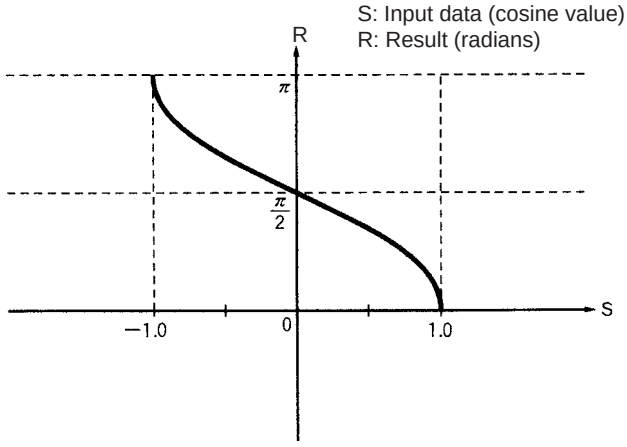
ACOS(464) computes the angle (in radians) for a cosine value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



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	ON if the result is negative. OFF in all other cases.

Precautions

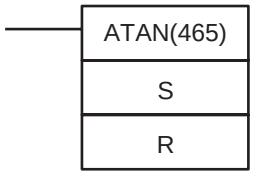
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-16 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

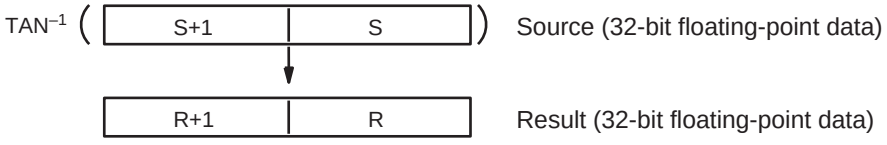
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

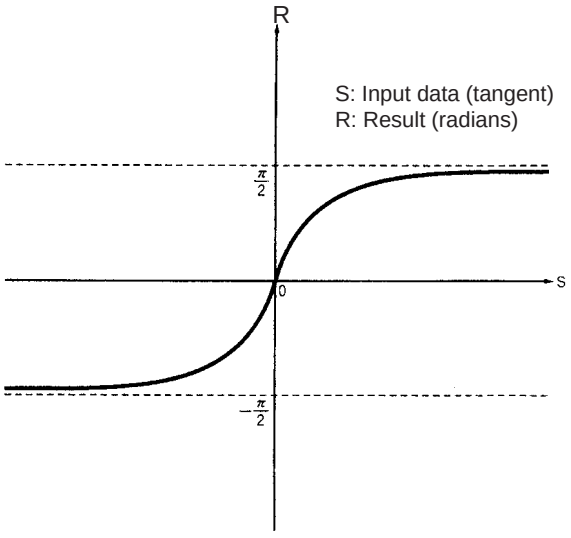
ATAN(465) computes the angle (in radians) for a tangent value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R.

(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

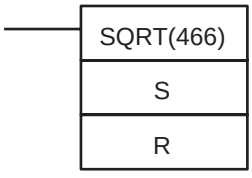
The source data in S+1 and S must be in IEEE754 floating-point data format.

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



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SQRT(466)
	Executed Once for Upward Differentiation	@SQRT(466)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

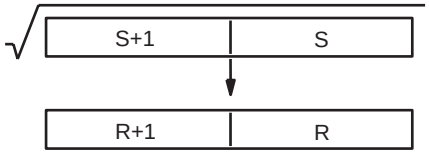
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.)



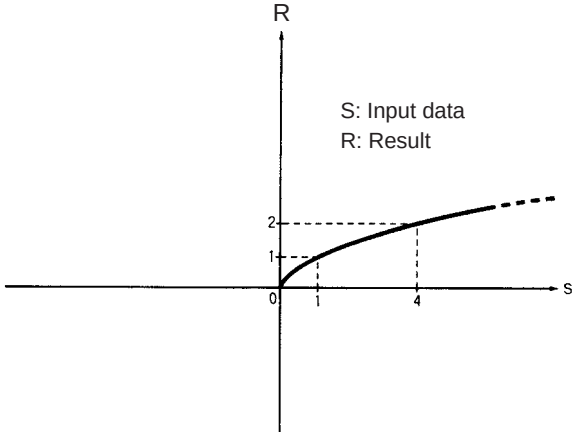
Source (32-bit floating-point data)

Result (32-bit floating-point data)

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

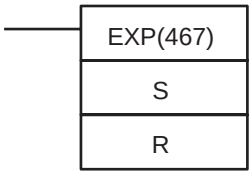
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

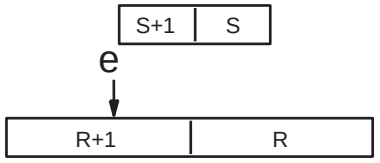
Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to 4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	

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 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)

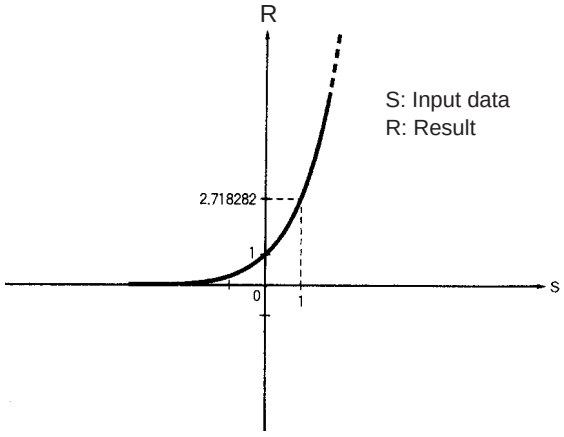
Result (32-bit floating-point data)

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.

Name	Label	Operation
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

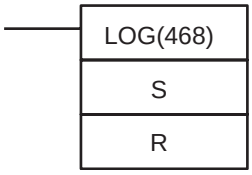
The source data in S+1 and S must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

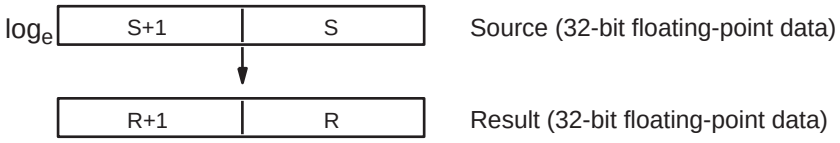
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.

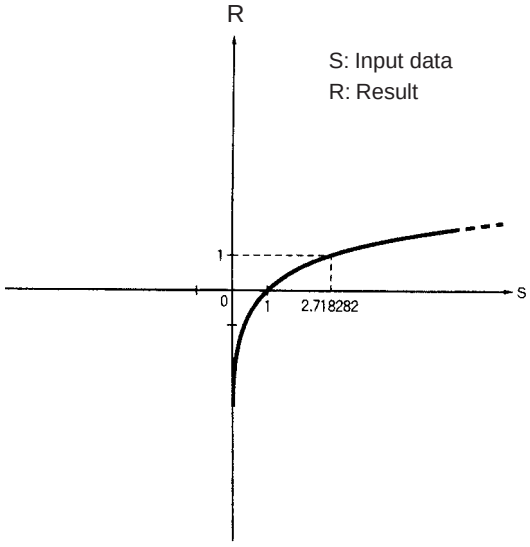


The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

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

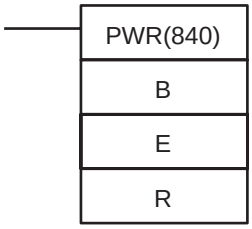
The source data in S+1 and S must be in IEEE754 floating-point data format.

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



B: First base word

E: First exponent word

R: First result word

Variations

Variations	**Executed Each Cycle for ON Condition**	PWR(840)
	Executed Once for Upward Differentiation	@PWR(840)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

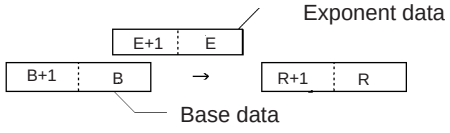
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	B	E	R
CIO Area	CIO 0 to CIO 6142		
Work Area	W0 to W510		
Holding Bit Area	H0 to H510		
Auxiliary Bit Area	A0 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D0 to D32766		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.
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

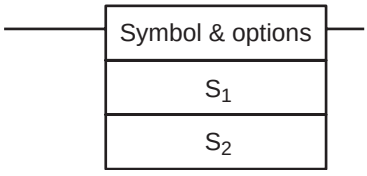
The base (B+1 and B) and the exponent (E+1 and E) must be in IEEE754 floating-point data format.

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.

Ladder Symbol



Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction
Immediate Refreshing Specification		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 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	
Data Registers	---	
Index Registers	IR0 to IR15 (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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

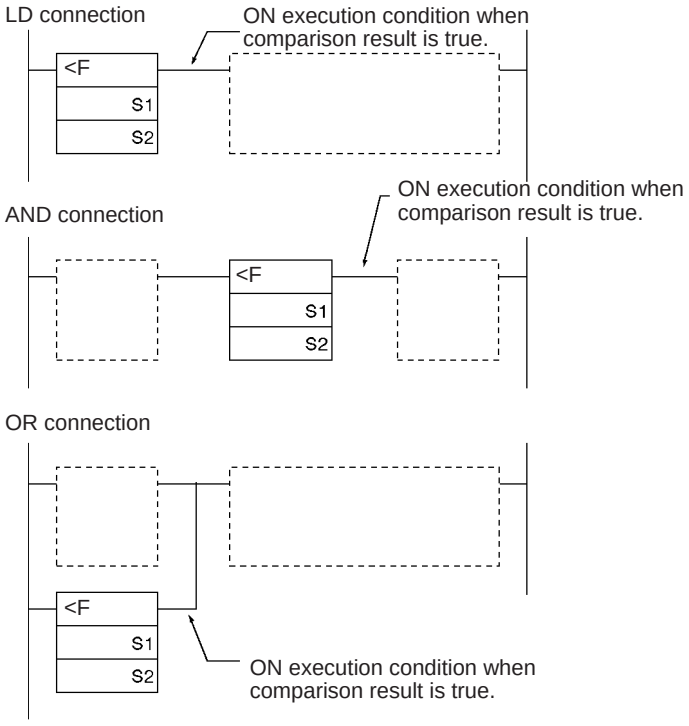
Description

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.

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	

Code	Mnemonic	Name	Function
333	LD>F	LOAD FLOATING GREATER THAN	True if C1 > C2
	AND>F	AND FLOATING GREATER THAN	
	OR>F	OR FLOATING GREATER THAN	
334	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 $+\infty$. ON if S_1+1, S_1 or S_2+1, S_2 is $-\infty$. 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 $(S_1+1, S_1) \geq (S_2+1, S_2)$. 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.
Negative Flag	N	Unchanged

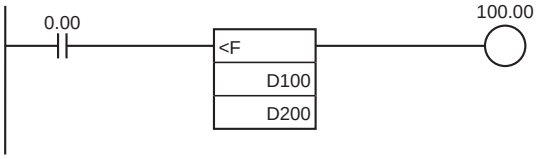
Precautions

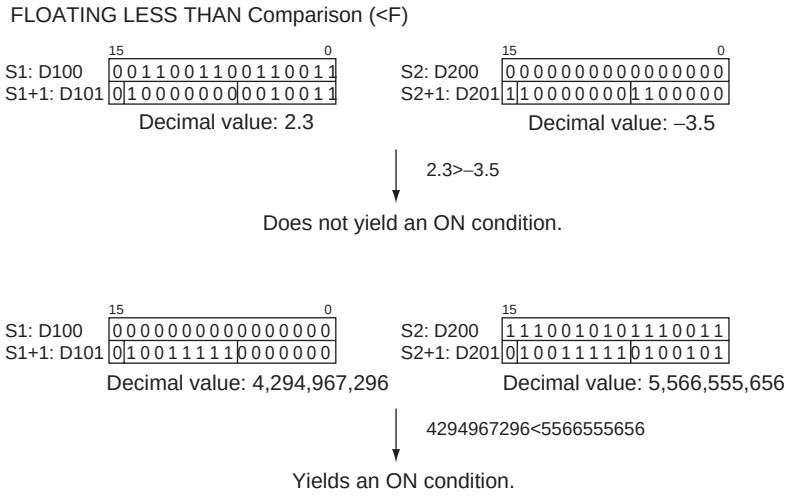
Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

Example

AND FLOATING LESS THAN: AND<F(331)

When CIO 0.00 is ON in the following example, the floating point data in D100, D101 is compared to the floating point data in D200, D201. If the content of D100, D101 is less than that of D200, D201, execution proceeds to the next line and CIO 100.00 is turned ON. If the content of D100, D101 is not less than that of D200, D201, execution does not proceed to the next instruction line.



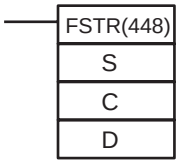


3-14-22 FLOATING-POINT TO ASCII: FSTR(448)

Purpose

Expresses a 32-bit floating-point value (IEEE754-format) in standard decimal notation or scientific notation and converts that value to ASCII text.

Ladder Symbol



S: First source word
C: First control word
D: First destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	FSTR(448)
	Executed Once for Upward Differentiation	@FSTR(448)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

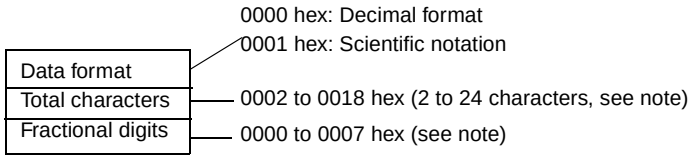
Area	S	C	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6141	CIO 0 to CIO 6143
Work Area	W0 to W510	W0 to W509	W0 to W511
Holding Bit Area	H0 to H510	H0 to H509	H0 to H511
Auxiliary Bit Area	A0 to A958	A0 to A957	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4093	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4093	C0000 to C4095
DM Area	D0 to D32766	D0 to D32765	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767	@ D0 to @ D32767	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767	*D0 to *D32767	*D0 to *D32767
Constants	#00000000 to #FFFFFFFF (binary)	---	
Data Registers	---		

Area	S	C	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –()IR15 ,IR0 to ,IR15		

Description

FSTR(448) expresses the 32-bit floating-point number in S+1 and S (IEEE754-format) in decimal notation or scientific notation according to the control data in words C to C+2, converts the number to ASCII text, and outputs the result to the destination words starting at D.

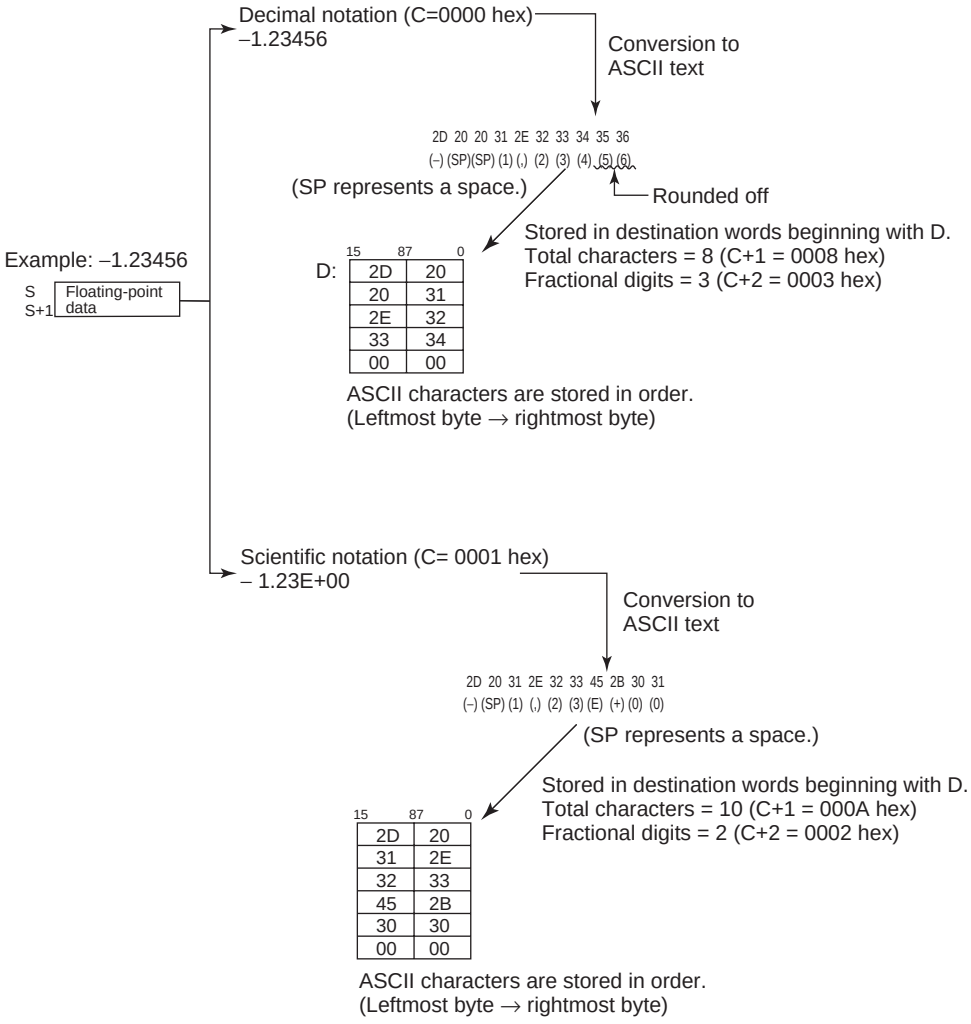
The following diagram shows the contents of the 3 control words.



Note: There are limits on the total number of characters and the number of fractional digits. See *Limits on the Number of ASCII Characters* on page 520 for details.

- The content of C (Data format) specifies whether to express the number in S+1, S in decimal notation or scientific notation.
  - Decimal notation
    Expresses a real number as an integer and fractional part.
    Example: 124.56
  - Scientific notation
    Expresses a real number as an integer part, fractional part, and exponent part.
    Example: 1.2456E-2 (1.2456×10^{-2})
- The content of C+1 (Total characters) specifies the number of ASCII characters after conversion including the sign symbol, numbers, decimal point and spaces.
- The content of C+2 (Fractional digits) specifies the number of digits (characters) below the decimal point.

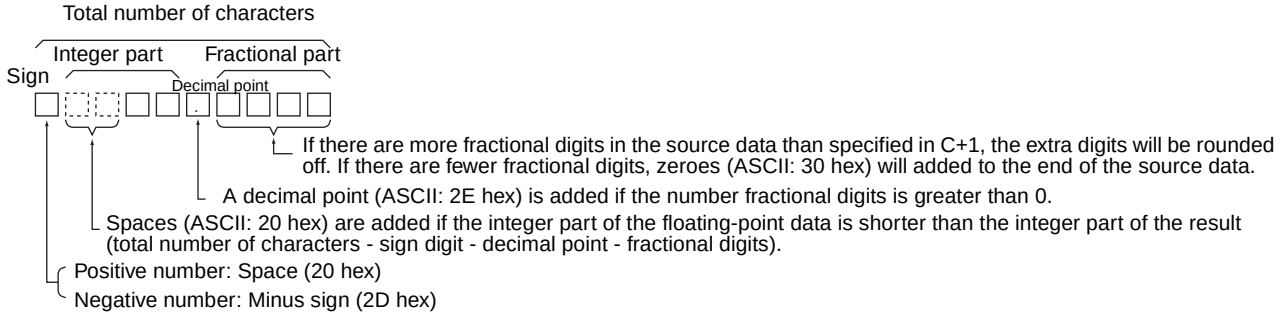
The ASCII text is stored in D and subsequent words in the following order: leftmost byte of D, rightmost byte of D, leftmost byte of D+1, rightmost byte of D+1, etc.



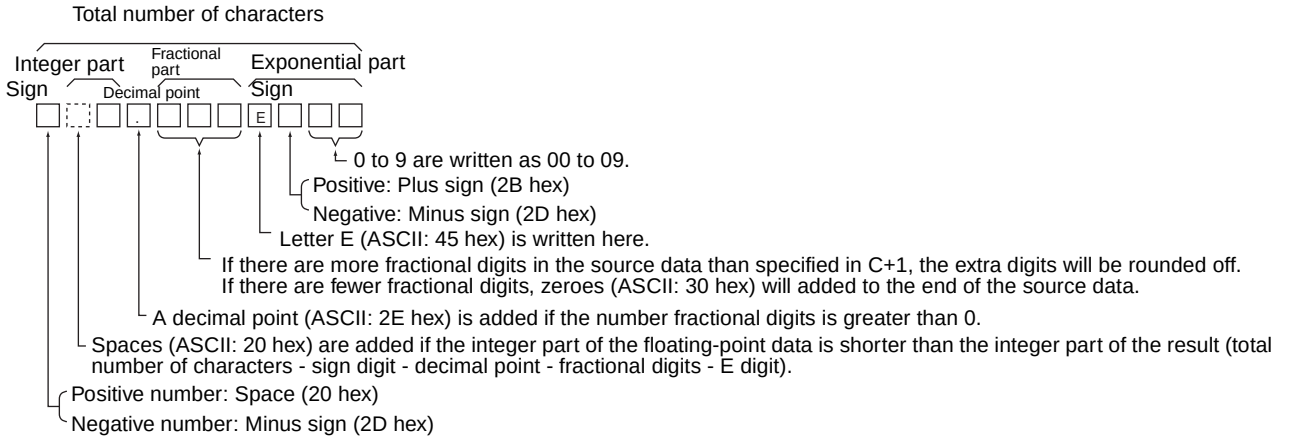
Storage of ASCII Text

After the floating-point number is converted to ASCII text, the ASCII characters are stored in the destination words beginning with D, as shown in the following diagrams. Different storage methods are used for decimal notation and scientific notation.

Decimal notation (C=0000 hex)



Scientific notation (C=0001 hex)



Note Either one or two bytes of zeroes are added to the end of ASCII text as an end code.
Total number of characters odd: 00 hex is stored after the ASCII text.
Total number of characters even: 0000 hex is stored after the ASCII text.

Limits on the Number of ASCII Characters

There are limits on the number of ASCII characters in the converted number. The Error Flag will be turned ON if the number of characters exceeds the maximum allowed.

1. Limits on the Total Number of ASCII Characters
   a. Decimal Notation (C = 0000 hex)
      - When there is no fractional part (C+2 = 0000 hex):
        $2 \leq$ Total Characters ≤ 24
      - When there is a fractional part (C+2 = 0001 to 0007 hex):
        (Fractional digits + 3) $\leq$ Total Characters ≤ 24

   b. Scientific Notation (C = 0001 hex)
      - When there is no fractional part (C+2 = 0000 hex):
        $6 \leq$ Total Characters ≤ 24
      - When there is a fractional part (C+2 = 0001 to 0007 hex):
        (Fractional digits + 7) $\leq$ Total Characters ≤ 24
2. Limits on the Number of Digits in the Integer Part
   a. Decimal Notation (C = 0000 hex)
      - When there is no fractional part (C+2 = 0000 hex):
        $1 \leq$ Number of Integer Digits ≤ 24
      - When there is a fractional part (C+2 = 0001 to 0007 hex):
        $1 \leq$ Number of Integer Digits $\leq (24 - \text{Fractional digits} - 2)$

   b. Scientific Notation (C = 0001 hex)
      1 digit (fixed)
3. Limits on the Number of Digits in the Fractional Part
   a. Decimal Notation (C = 0000 hex)
      - Fractional Digits ≤ 7
      - Also: Fractional Digits $\leq$ (Total Number of ASCII Characters $- 3$)

   b. Scientific Notation (C = 0001 hex)
      - Fractional Digits ≤ 7
      - Also: Fractional Digits $\leq$ (Total Number of ASCII Characters $- 3$)

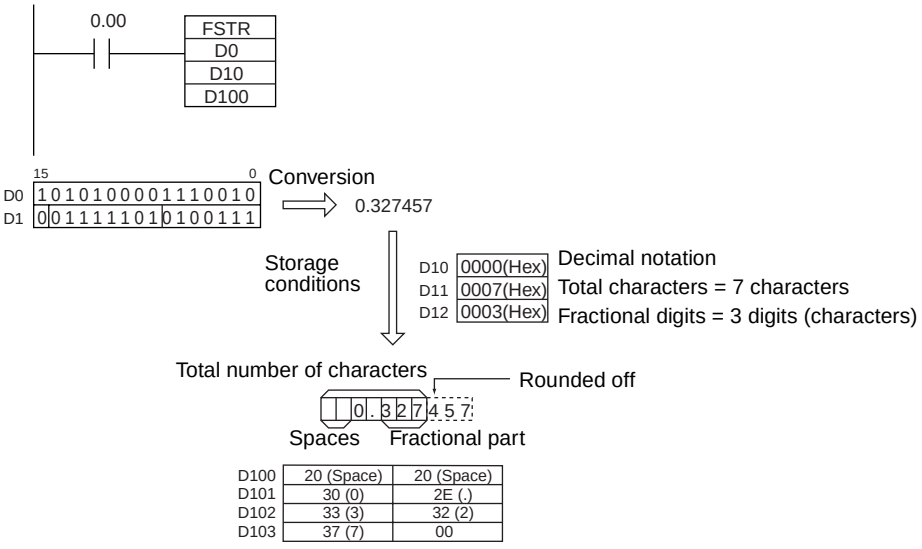
Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not a valid floating-point number (NaN). ON if the data in S+1 and S is +∞ or −∞. ON if the Data Format setting in C is not 0000 or 0001. ON if the Total Characters setting in C+1 is not within the allowed range. (See *1. Limits on the Total Number of ASCII Characters* above for details.) ON if the Fractional Digits setting in C+2 is not within the allowed range. (See *3. Limits on the Number of Digits in the Fractional Part* above for details.) OFF in all other cases.
Equals Flag	=	ON if the conversion result is 0. OFF in all other cases.

Examples

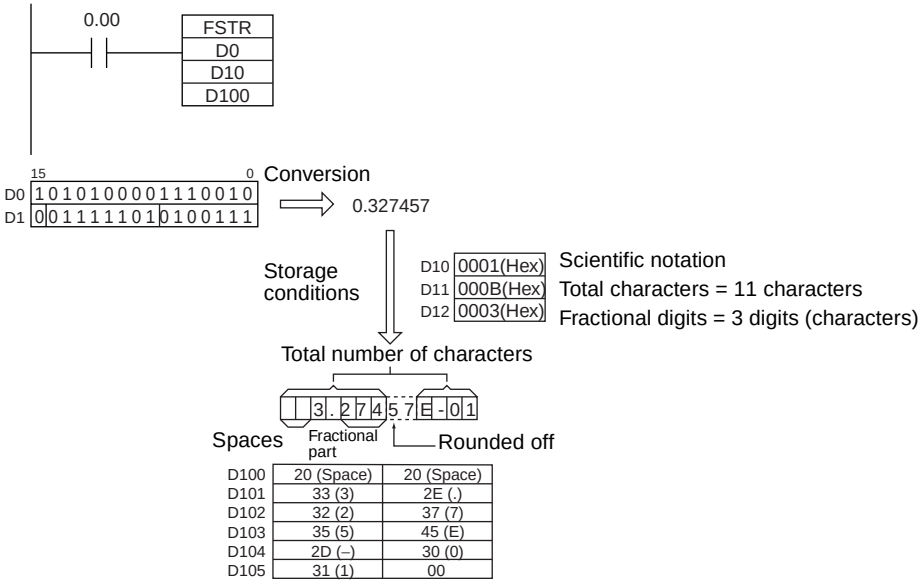
Converting to ASCII Text in Decimal Notation

When CIO 0.00 is ON in the following example, FSTR(448) converts the floating-point data in D1 and D0 to decimal-notation ASCII text and writes the ASCII text to the destination words beginning with D100. The contents of the control words (D10 to D12) specify the details on the data format (decimal notation, 7 characters total, 3 fractional digits).



Converting to ASCII Text in Scientific Notation

When CIO 0.00 is ON in the following example, FSTR(448) converts the floating-point data in D0 to scientific-notation ASCII text and writes the ASCII text to the destination words beginning with D100. The contents of the control words (D10 to D12) specify the details on the data format (scientific notation, 11 characters total, 3 fractional digits).

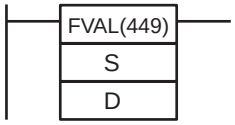


3-14-23 ASCII TO FLOATING-POINT: FVAL(449)

Purpose

Converts a number expressed in ASCII text (decimal or scientific notation) to a 32-bit floating-point value (IEEE754-format) and outputs the floating-point value to the specified words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	FVAL(449)
	Executed Once for Upward Differentiation	@FVAL(449)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 0 to CIO 6143	CIO 0 to CIO 6142
Work Area	W0 to W511	W0 to W510
Holding Bit Area	H0 to H511	H0 to H510
Auxiliary Bit Area	A0 to A959	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4095	C0000 to C4094
DM Area	D0 to D32767	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	

Area	S	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –()IR15 ,IR0 to ,IR15	

Description

FVAL(449) converts the specified ASCII text number (starting at word S) to a 32-bit floating-point number (IEEE754-format) and outputs the result to the destination words starting at D.

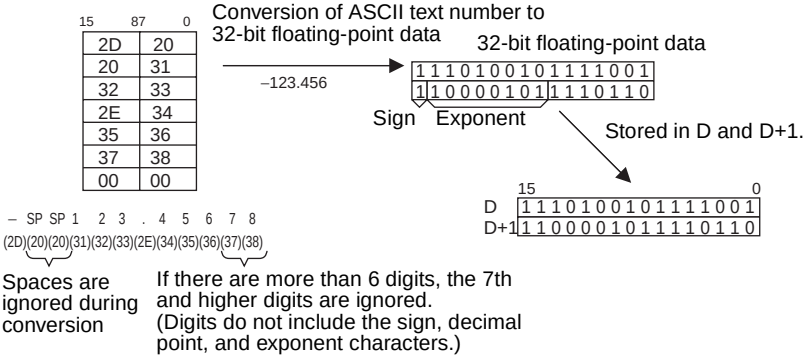
FVAL(449) can convert ASCII text in decimal or scientific notation if it meets the following conditions:

- Decimal notation
  Real numbers expressed with an integer and fractional part.
  Example: 124.56
- Scientific notation
  Real numbers expressed as an integer part, fractional part, and exponent part.
  Example: 1.2456E-2 (1.2456×10^{-2})

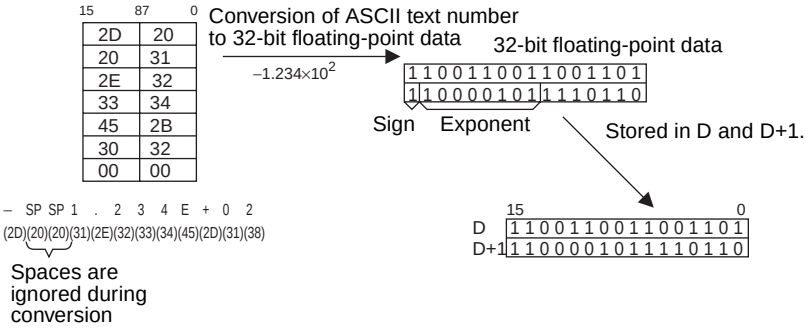
The data format (decimal or scientific notation) is detected automatically.

The ASCII text must be stored in S and subsequent words in the following order: leftmost byte of S, rightmost byte of S, leftmost byte of S+1, rightmost byte of S+1, etc.

Decimal notation



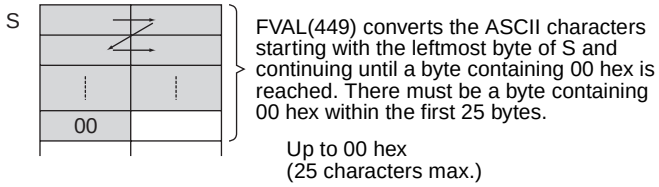
Scientific notation



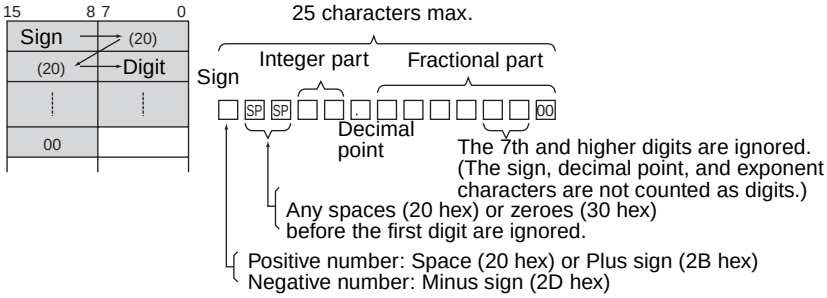
Storage of ASCII Text

The following diagrams show how the ASCII text number is converted to floating-point data. Different conversion methods are used for numbers stored with decimal notation and scientific notation.

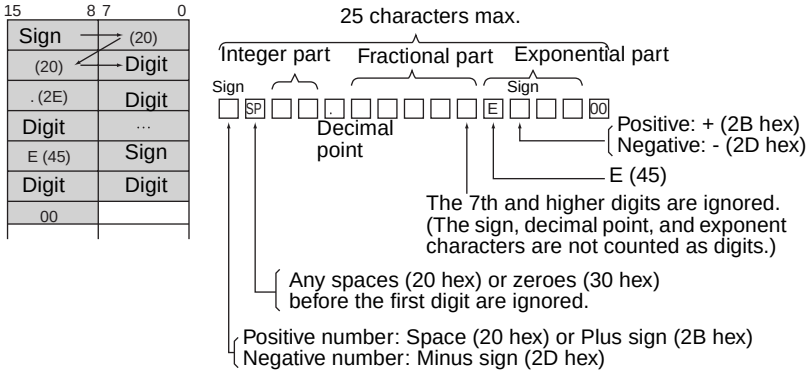
ASCII Character Storage



Decimal notation



Scientific notation



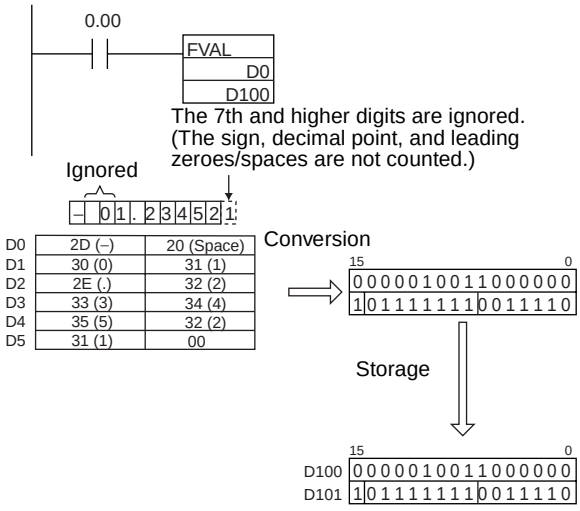
Flags

Name	Label	Operation
Error Flag	ER	ON if the digits (integer and fractional parts) in the source data starting at S are not 30 to 39 hex (0 to 9). ON if the first two digits of the exponential part do not contain 45 and 2B hex (E+) or 45 and 2D hex (E-). (integer and fractional parts) in the source data starting at S are not 30 to 39 hex (0 to 9). ON if there are two or more exponential parts in the source data. ON if the data is +∞ or –∞ after conversion. ON is the are 0 characters in the text data. ON if a byte containing 00 hex is not found within the first 25 characters. OFF in all other cases.
Equals Flag	=	ON if the conversion result is 0. OFF in all other cases.

Examples

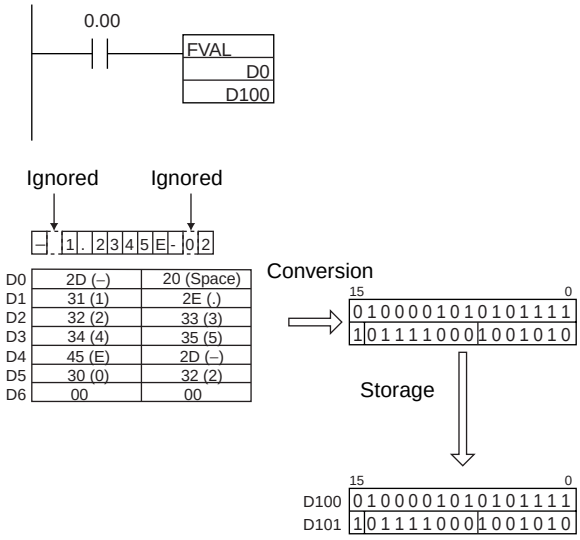
Converting ASCII Text in Decimal Notation to Floating-point Data

When CIO 0.00 is ON in the following example, FVAL(449) converts the specified decimal-notation ASCII text number in the source words starting at D0 to floating-point data and writes the result to destination words D100 and D101.



Converting ASCII Text in Scientific Notation

When CIO 0.00 is ON in the following example, FVAL(449) converts the specified scientific-notation ASCII text number in the source words starting at D0 to floating-point data and writes the result to destination words D100 and D101.



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.

Instruction	Mnemonic	Function code	Page
DOUBLE FLOATING TO 16-BIT	FIXD	841	531
DOUBLE FLOATING TO 32-BIT	FIXLD	842	533
16-BIT TO DOUBLE FLOATING	DBL	843	534
32-BIT TO DOUBLE FLOATING	DBLL	844	535
DOUBLE FLOATING-POINT ADD	+D	845	537
DOUBLE FLOATING-POINT SUBTRACT	–D	846	539
DOUBLE FLOATING-POINT MULTIPLY	∗D	847	541
DOUBLE FLOATING-POINT DIVIDE	/D	848	543
DOUBLE DEGREES TO RADIANS	RADD	849	545
DOUBLE RADIANS TO DEGREES	DEGD	850	546
DOUBLE SINE	SIND	851	548
DOUBLE COSINE	COSD	852	549
DOUBLE TANGENT	TAND	853	551
DOUBLE ARC SINE	ASIND	854	552
DOUBLE ARC COSINE	ACOSD	855	554
DOUBLE ARC TANGENT	ATAND	856	556
DOUBLE SQUARE ROOT	SQRTD	857	558
DOUBLE EXPONENT	EXPD	858	559
DOUBLE LOGARITHM	LOGD	859	561
DOUBLE EXPONENTIAL POWER	PWRD	860	563
Double-precision Floating-point Symbol Comparison Instructions	LD, AND, OR + =D, <>D, <D, <=D, >D, or >=D	335 to 340	564

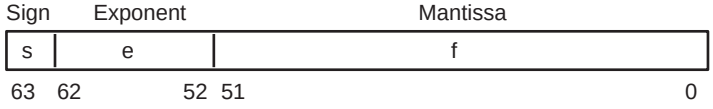
Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

$$\text{Real number} = (-1)^{s}\ 2^{e-1,023}\ (1.f)$$

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

Data	No. of bits	Contents
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.

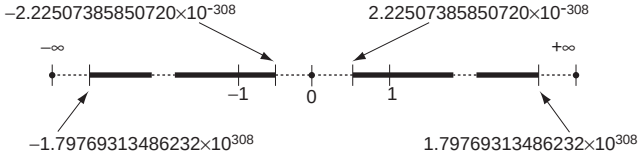
Number of Digits

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:

- $-\infty$
- $-1.79769313486232 \times 10^{308} \leq \text{value} \leq -2.22507385850720 \times 10^{-308}$
- 0
- $2.22507385850720 \times 10^{-308} \leq \text{value} \leq 1.79769313486232 \times 10^{30}$
- $+\infty$
- Not a number (NaN)



Special Numbers

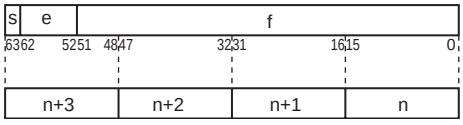
The formats for NaN, ±∞, and 0 are as follows:

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.

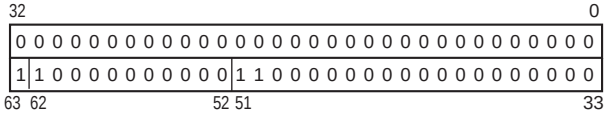
The exponent (e) will be expressed from 1 to 2,046, and the real exponent will be 1,023 less, i.e., –1,022 to 1,023.

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.

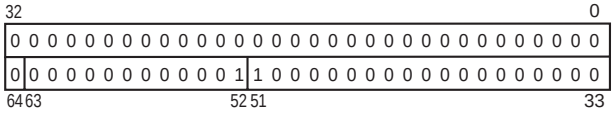
The exponent (e) will be 0, and the real exponent will be –1,022.

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 –∞.

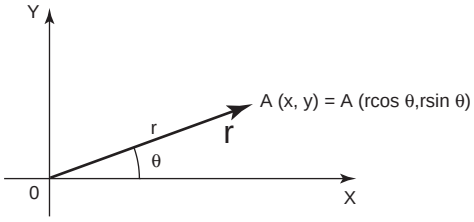
The Equals Flag will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow Flag will turn ON.

Comparing Single-precision and Double-precision Calculations

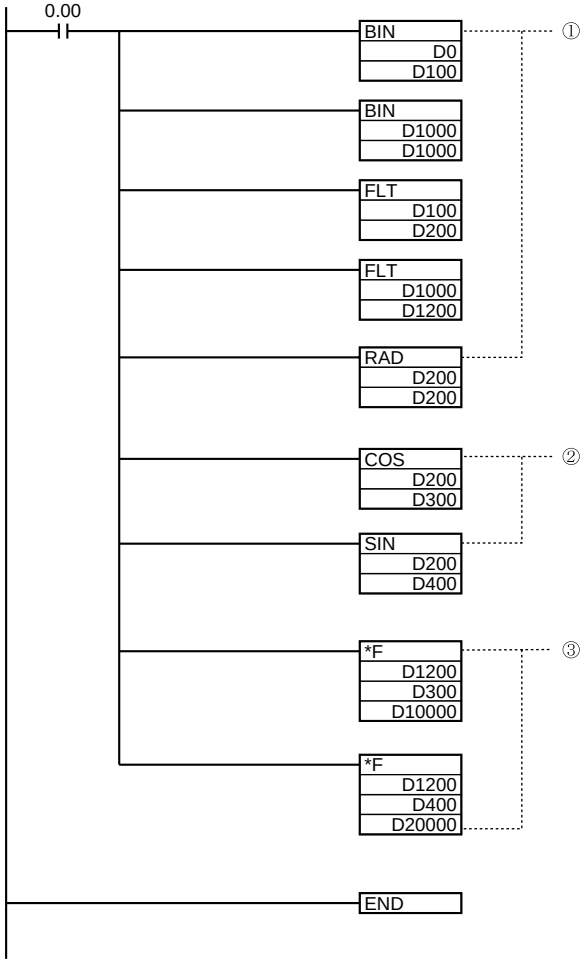
This example shows the differences in between single-precision and double-precision calculations when the following vector expressed in polar coordinates is converted to rectangular coordinates A (x,y).

$$r = re^{j\left(\frac{\pi}{360}\right)\theta}$$

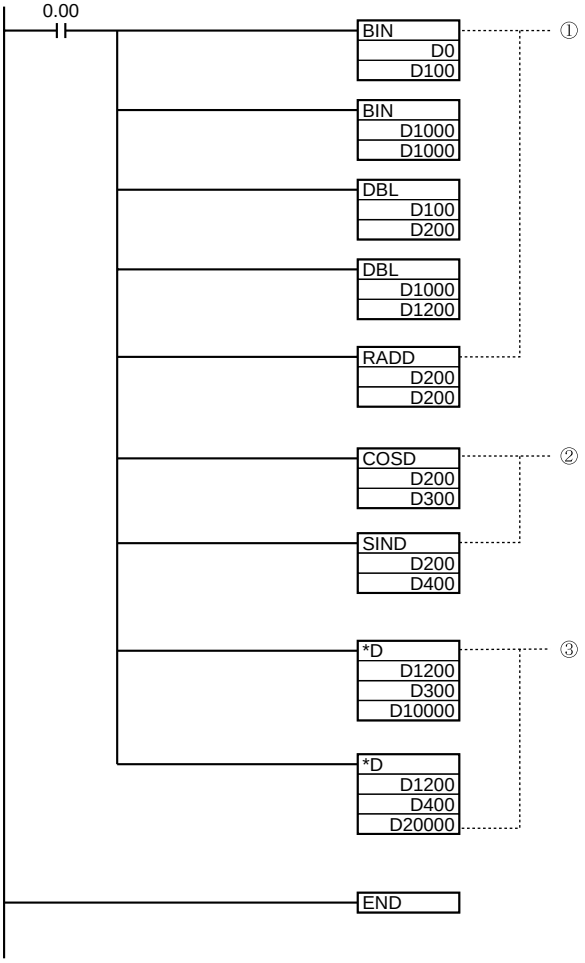
In this example, the 4-digit BCD angle (θ, in degrees) is read from D0 and the 4-digit BCD distance (r) is read from D1000.



- 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 D200 and D201.
   c. RAD(458) converts the angle data in D200 and D201 to radians.
   d. The floating-point data for the radius r is output to D1200 and D1201.
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 D300 and D301.
   b. The value for sin θ is output to D400 and D401.
3. This program section calculates x (r × cos θ) and y (r × sin θ).
   a. The value for x (r × cos θ) is output to D10000 and D10001.
   b. The value for y (r × sin θ) is output to D20000 and D20001.

Coordinate	Floating-point number	Real number
x	4116 59CF	3.4202015399933
y	405A E495	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 D200 to D203.
   c. RADD(849) converts the angle data in words D200 to D203 to radians.
   d. The floating-point data for the radius r is output to words D1200 to D1203.
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 D300 to D303.
   b. The value for sin θ is output to words D400 and D403.
3. This program section calculates x (r × cos θ) and y (r × sin θ).
   a. The value for x (r × cos θ) is output to words D10000 to D10003.
   b. The value for y (r × sin θ) is output to D20000 and D20003.

Coordinate	Floating-point number	Real number
x	4022 CB39 E973 5C32	3.4202014332567
y	400B 5C92 91AC 8EEB	9.3969262078591

Comparison of the Calculation Results

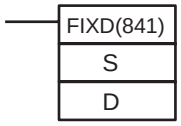
When the real-number results are compared, it is clear that the double-precision calculation yields a more accurate result.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

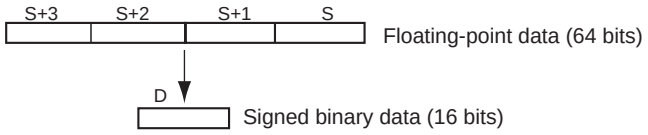
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	CIO 0 to CIO 6143
Work Area	W0 to W508	W0 to W511
Holding Bit Area	H0 to H508	H0 to H511
Auxiliary Bit Area	A0 to A956	A448 to A959
Timer Area	T0000 to T4092	T0000 to T4095
Counter Area	C0000 to C4092	C0000 to C4095
DM Area	D0 to D32764	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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

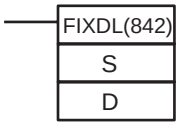
Name	Label	Operation
Error Flag	ER	ON if the source data (S to S+3) is not a number (NaN). ON if the integer portion of the source data (S to S+3) is not within the range of –32,768 to 32,767. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

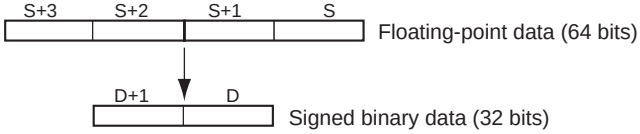
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	CIO 0 to CIO 6142
Work Area	W0 to W508	W0 to W510
Holding Bit Area	H0 to H508	H0 to H510
Auxiliary Bit Area	A0 to A956	A448 to A958
Timer Area	T0000 to T4092	T0000 to T4094
Counter Area	C0000 to C4092	C0000 to C4094
DM Area	D0 to D32764	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.
A floating-point value of –2,147,483,640.5 is converted to –2,147,483,640.

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

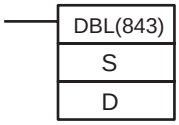
The content of words S to S+3 must be floating-point data and the integer portion must be in the range of –2,147,483,648 to 2,147,483,647.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

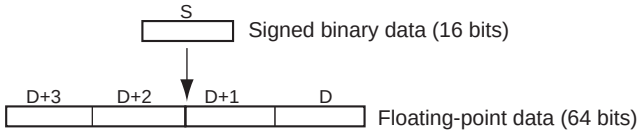
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6140
Work Area	W0 to W511	W0 to W508
Holding Bit Area	H0 to H511	H0 to H508
Auxiliary Bit Area	A0 to A959	A448 to A956
Timer Area	T0000 to T4095	T0000 to T4092
Counter Area	C0000 to C4095	C0000 to C4092
DM Area	D0 to D32767	D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---

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 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

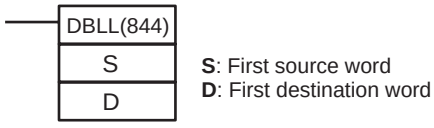
The content of S must contain signed binary data with a (decimal) value in the range of –32,768 to 32,767.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

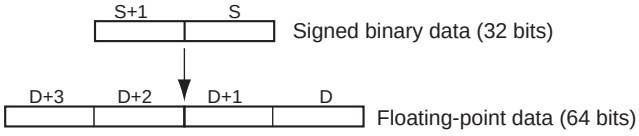
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6140
Work Area	W0 to W510	W0 to W508
Holding Bit Area	H0 to H510	H0 to H508
Auxiliary Bit Area	A0 to A958	A448 to A956
Timer Area	T0000 to T4094	T0000 to T4092
Counter Area	C0000 to C4094	C0000 to C4092
DM Area	D0 to D32766	D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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

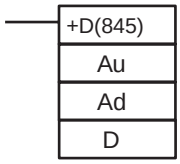
The result will not be exact if a number with an absolute value greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted.

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.
Immediate Refreshing Specification		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 0 to CIO 6140		
Work Area	W0 to W508		
Holding Bit Area	H0 to H508		
Auxiliary Bit Area	A0 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D0 to D32764		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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	+∞	–∞	See note 2.
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

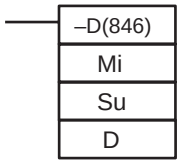
The augend (Au to Au+3) and Addend (Ad to Ad+3) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

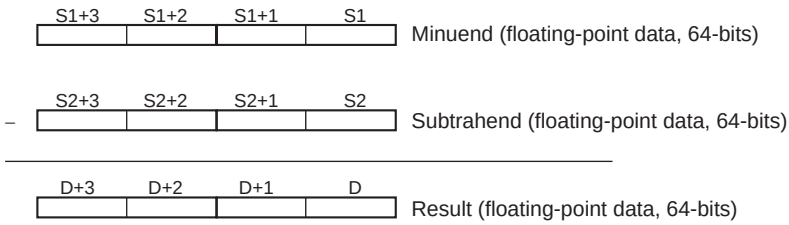
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	D
CIO Area	CIO 0 to CIO 6140		
Work Area	W0 to W508		
Holding Bit Area	H0 to H508		
Auxiliary Bit Area	A0 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D0 to D32764		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

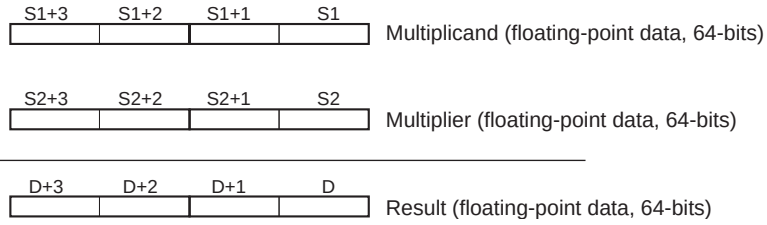
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	D
CIO Area	CIO 0 to CIO 6140		
Work Area	W0 to W508		
Holding Bit Area	H0 to H508		
Auxiliary Bit Area	A0 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D0 to D32764		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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

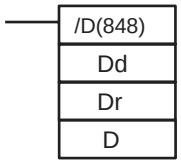
The Multiplicand (Md to Md+3) and Multiplier (Mr to Mr+3) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

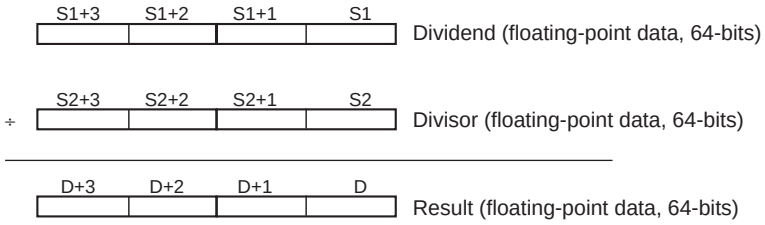
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	D
CIO Area	CIO 0 to CIO 6140		
Work Area	W0 to W508		
Holding Bit Area	H0 to H508		
Auxiliary Bit Area	A0 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D0 to D32764		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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.
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

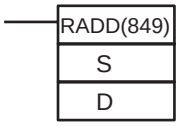
The Dividend (Dd to Dd+3) and Divisor (Dr to Dr+3) data must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

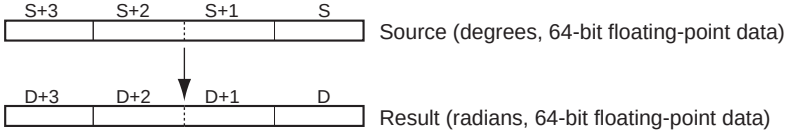
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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 $\times$ π/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 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

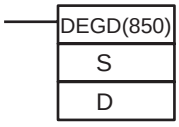
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

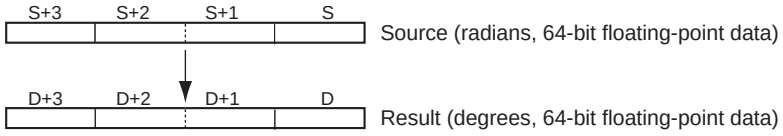
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	

Area	S	D
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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

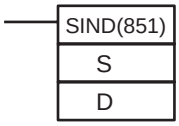
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

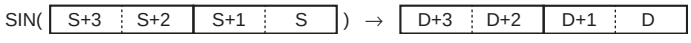
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

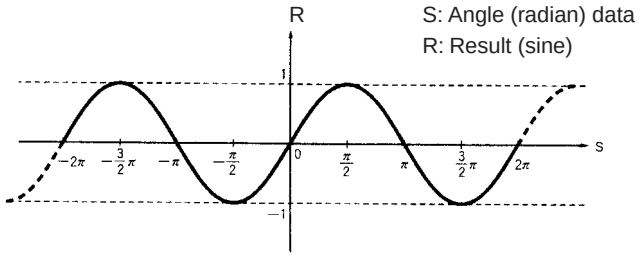
Description

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

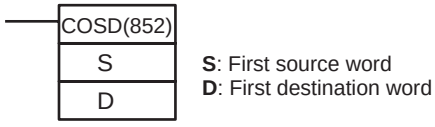
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

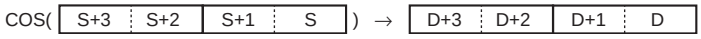
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	

Area	S	D
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

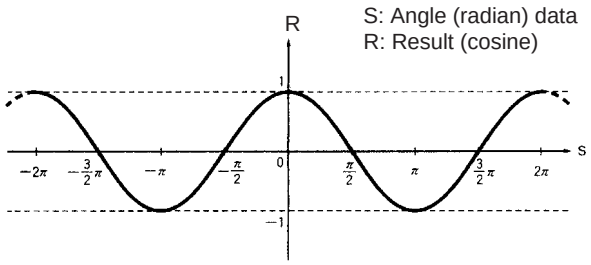
Description

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

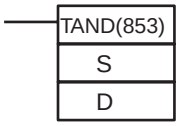
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

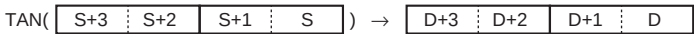
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

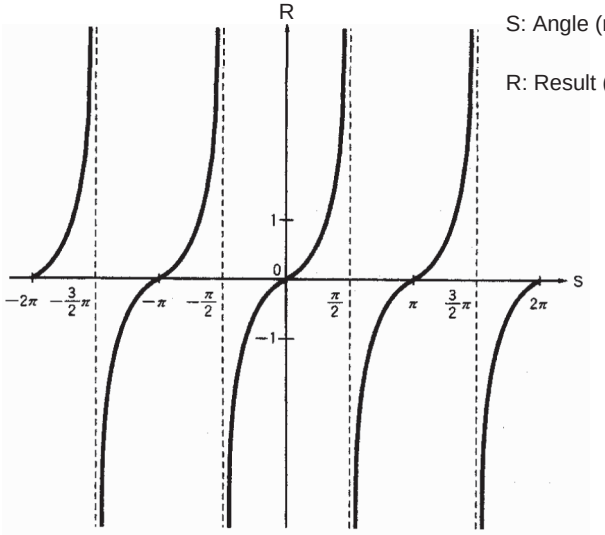
TAND(853) calculates the tangent of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



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.



S: Angle (radian) data

R: Result (tangent)

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

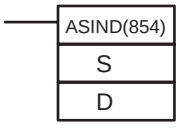
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

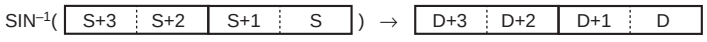
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

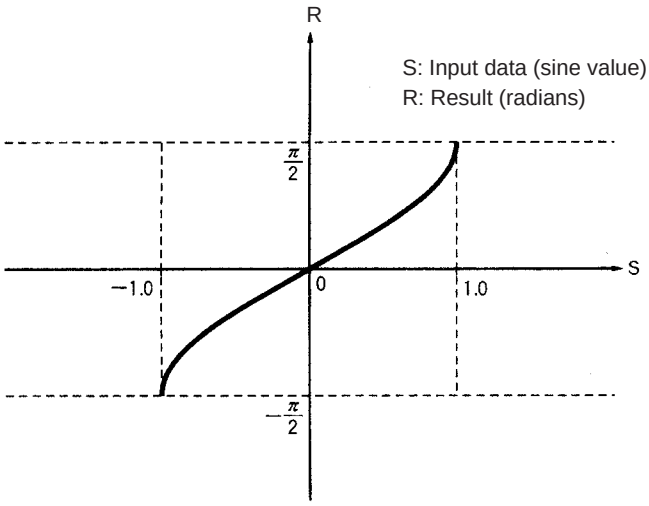
ASIND(854) computes the angle (in radians) for a sine value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



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

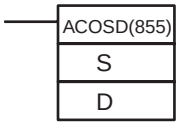
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		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 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

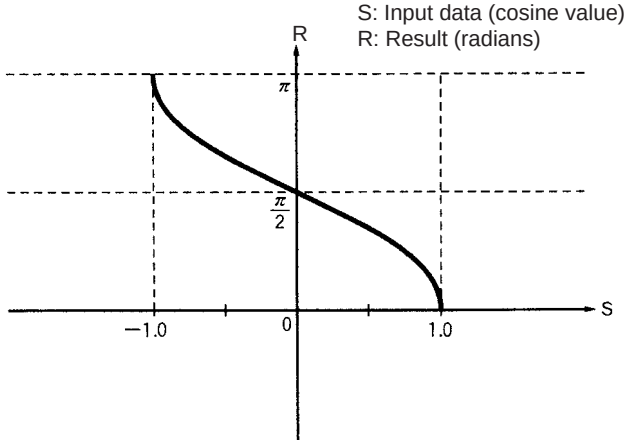
ACOSD(855) computes the angle (in radians) for a cosine value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)

$\cos^{-1}$(S+3 | S+2 | S+1 | S) → D+3 | D+2 | D+1 | D

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

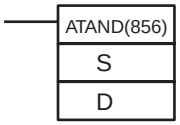
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		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 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

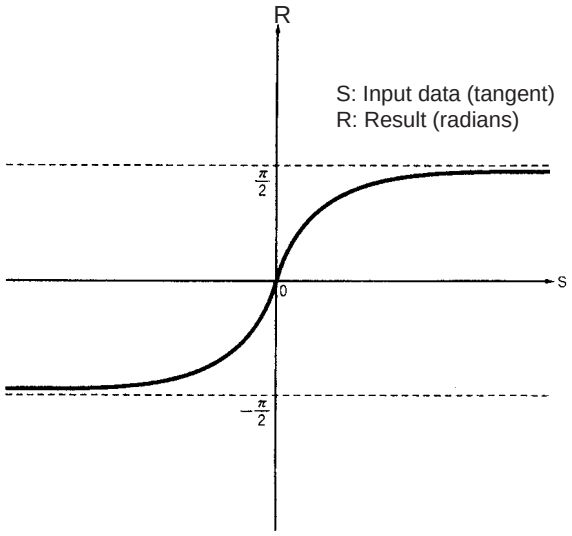
Description

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.)

TAN^{-1}(S+3 | S+2 | S+1 | S) → D+3 | D+2 | D+1 | D

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

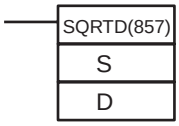
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

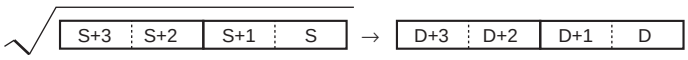
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

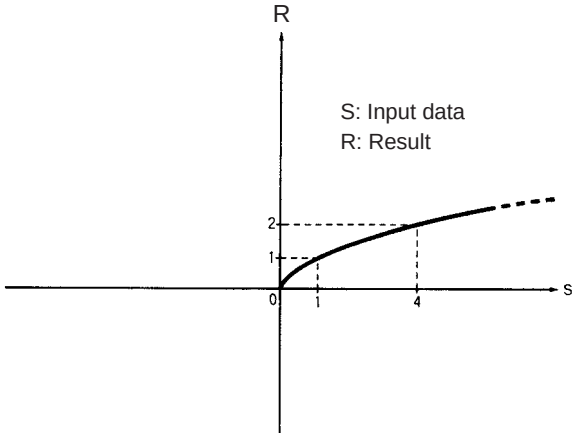
SQRTD(857) calculates the square root of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



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

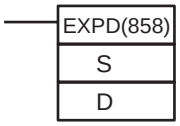
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

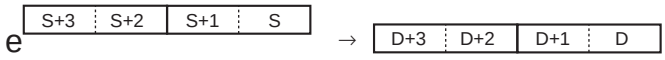
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.

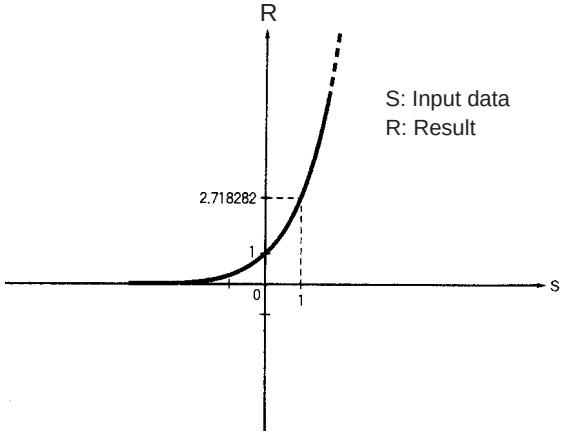


If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

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

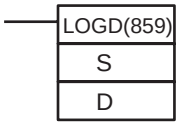
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		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 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 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.

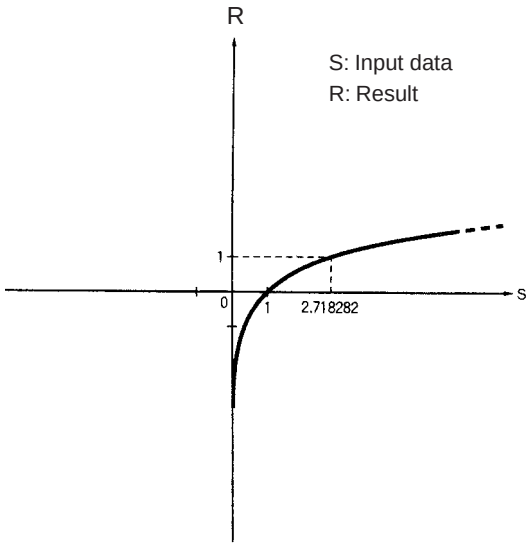
$\log_e$ [S+3 | S+2 | S+1 | S] → [D+3 | D+2 | D+1 | D]

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

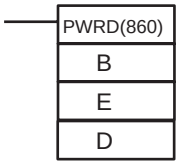
The source data in words S to S+3 must be in IEEE754 floating-point data format.

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

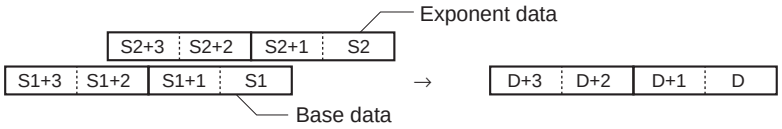
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	B	E	D
CIO Area	CIO 0 to CIO 6140		
Work Area	W0 to W508		
Holding Bit Area	H0 to H508		
Auxiliary Bit Area	A0 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D0 to D32764		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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

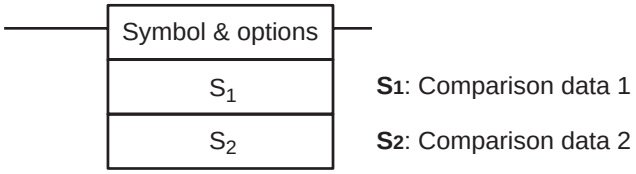
The base data (B to B+3) and the exponent data (E to E+3) must be in IEEE754 floating-point data format.

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.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction
Immediate Refreshing Specification		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 0 to CIO 6140	
Work Area	W0 to W508	
Holding Bit Area	H0 to H508	
Auxiliary Bit Area	A0 to A956	
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D0 to D32764	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

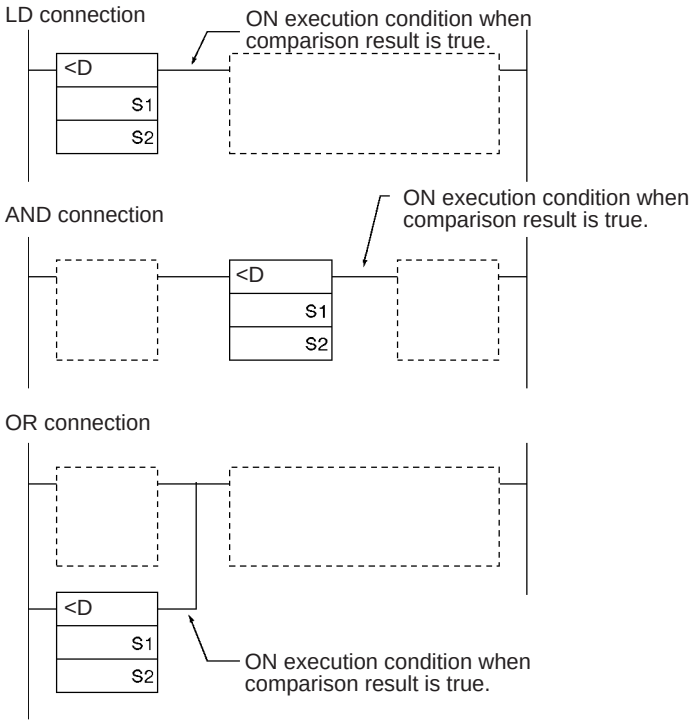
Description

The input comparison instructions compare 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

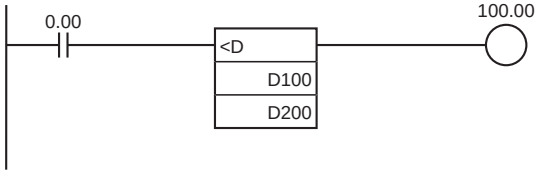
Precautions

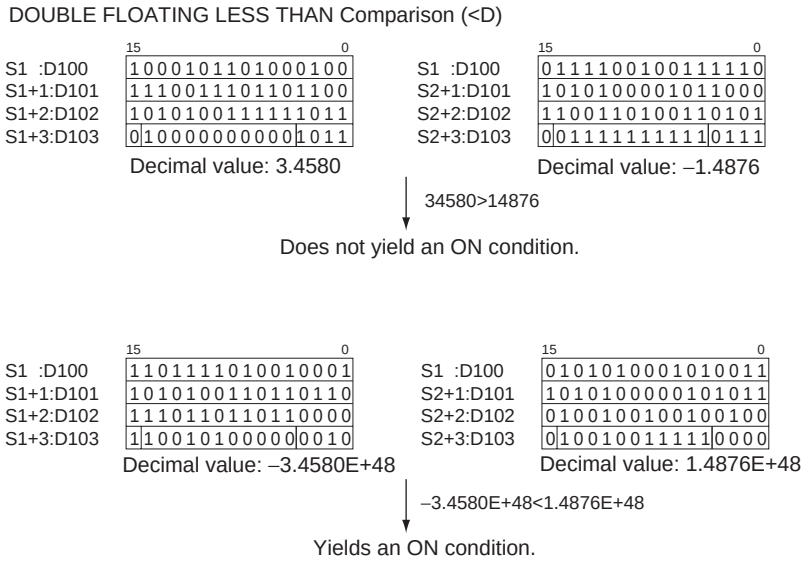
Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

Example

AND DOUBLE FLOATING LESS THAN: AND<D(331)

When CIO 0.00 is ON in the following example, the floating point data in words D100 to D103 is compared to the floating point data in words D200 to D203. If the content of D100 to D103 is less than that of D200 to D203, execution proceeds to the next line and CIO 100.00 is turned ON. If the content of D100 to D103 is not less than that of D200 to D203, execution does not proceed to the next instruction line.





3-16 Table Data Processing Instructions

This section describes instructions used to handle table data, stacks, and other ranges of data.

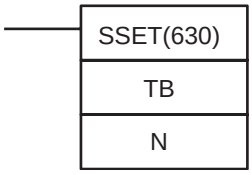
Instruction	Mnemonic	Function code	Page
SET STACK	SSET	630	568
PUSH ONTO STACK	PUSH	632	571
FIRST IN FIRST OUT	FIFO	633	574
LAST IN FIRST OUT	LIFO	634	576
DIMENSION RECORD TABLE	DIM	631	579
SET RECORD LOCATION	SETR	635	581
GET RECORD NUMBER	GETR	636	583
DATA SEARCH	SRCH	181	585
SWAP BYTES	SWAP	637	587
FIND MAXIMUM	MAX	182	589
FIND MINIMUM	MIN	183	592
SUM	SUM	184	595
FRAME CHECKSUM	FCS	180	598
STACK NUMBER OUTPUT	SNUM	638	601
STACK DATA READ	SREAD	639	604
STACK DATA OVERWRITE	SWRIT	640	607
STACK DATA INSERT	SINS	641	610
STACK DATA DELETE	SDEL	642	613

3-16-1 SET STACK: SSET(630)

Purpose

Defines a stack of the specified length beginning at the specified word.

Ladder Symbol



TB: First stack address

N: Number of words

Variations

Variations		
	Executed Each Cycle for ON Condition	SSET(630)
	Executed Once for Upward Differentiation	@SSET(630)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

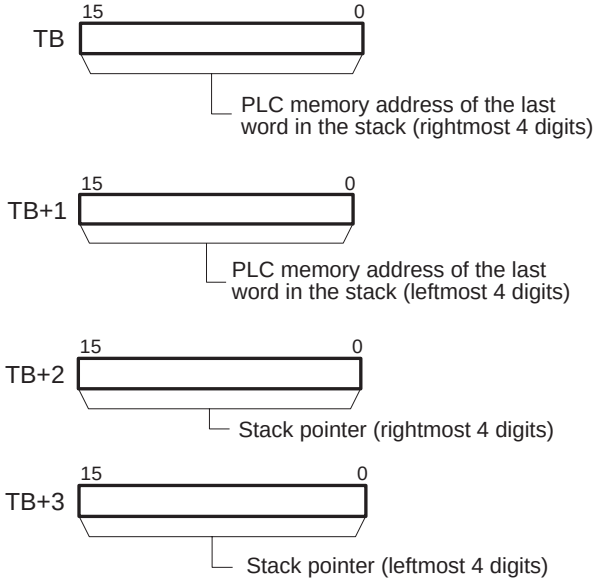
Applicable Program Areas

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

Operands

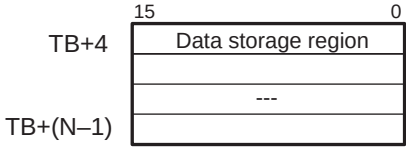
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



Note

(1) The initial value of the stack pointer is always the PLC memory address of TB+4.

(2) TB and TB+(N–1) must be in the same data area.

Operand Specifications

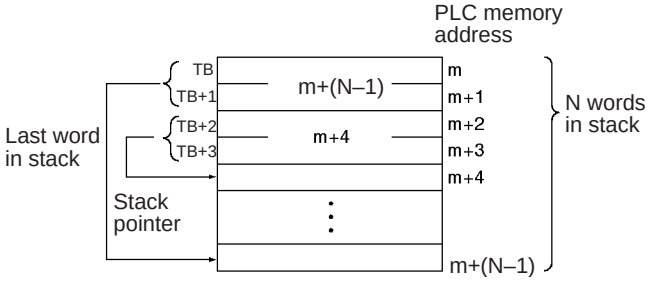
Area	TB	N
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	A0 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	

Area	TB	N
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	#0005 to #FFFF (binary) or &5 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SSET(630) secures a stack with N words beginning at TB and ending at TB+(N–1). The first two words of the stack (TB+1 and TB) contain the 8-digit hexadecimal PLC memory address of the last word in the stack. The next two words (TB+3 and TB+2) contain the stack pointer. The stack pointer is the PLC memory address of the next word in the stack that will be overwritten by PUSH(632); its initial value is the address of TB+4.

SSET(630) automatically initializes the data region of the stack (TB+4 through TB+(N–1)) to zeroes. The following diagram shows the basic structure of a stack.



SSET(630) just establishes and initializes a stack. Use the following instructions to store in the stack and read data from the stack.

1,2,3...

1. PUSH(632) stores data in the stack one word at a time.
2. FIFO(633) and LIFO(634) read data from the stack. FIFO(633) reads the first word that was stored; LIFO(634) reads the last word that was stored.
3. The stack pointer value in the stack control word is automatically updated when PUSH(632), FIFO(633), or LIFO(634) is executed. Normally, users need not be concerned about the stack control word. When accessing the contents of the stack other than by using the above instructions, set the stack pointer value using the Index Register (IR) for indirect referencing.

Flags

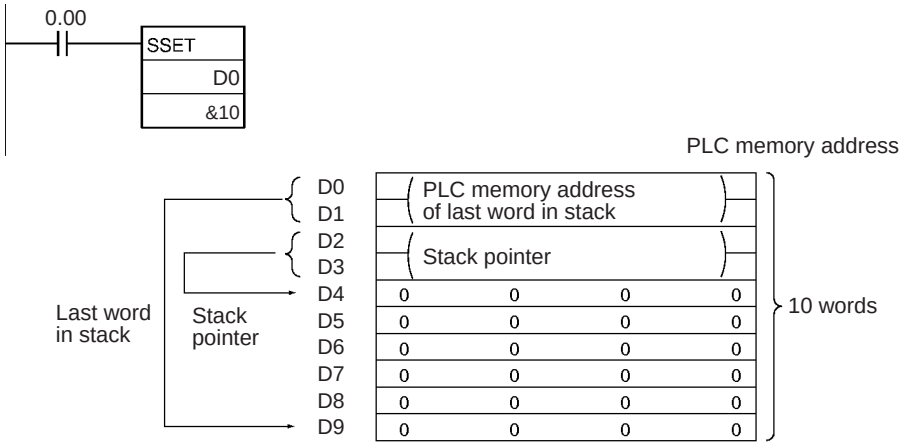
Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0005 to FFFF. OFF in all other cases.

Precautions

The minimum value for the number of words in the stack (N) is 5 because N includes the four words that contain the pointer to the last word in the stack and the stack pointer. An error will occur if N is not in the range 0005 to FFFF.

Examples

When CIO 0.00 is ON in the following example, SSET(630) secures a 10-word stack from D0 to D9. D0 and D1 contain the PLC memory address of the last word in the stack. D2 and D3 contain the stack pointer. The stack itself begins in D4.

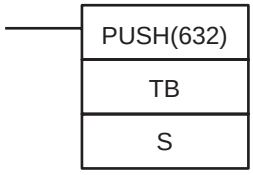


3-16-2 PUSH ONTO STACK: PUSH(632)

Purpose

Writes one word of data to the specified stack.

Ladder Symbol



TB: First stack address

S: Source word

Variations

Variations		
	Executed Each Cycle for ON Condition	PUSH(632)
	Executed Once for Upward Differentiation	@PUSH(632)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

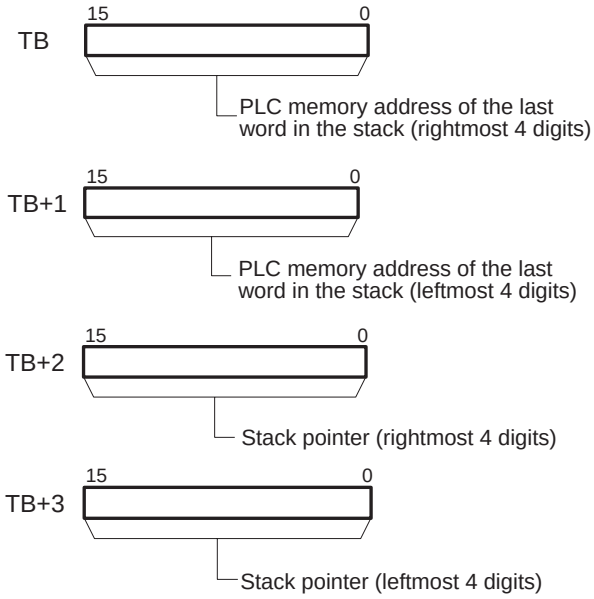
Applicable Program Areas

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

Operands

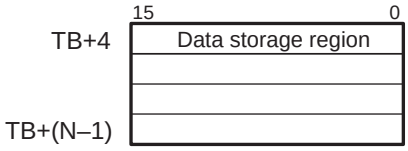
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

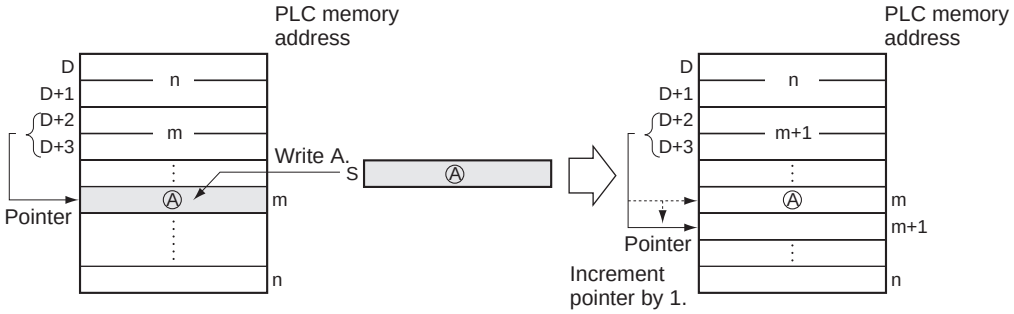


Operand Specifications

Area	TB	S
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	A0 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

PUSH(632) writes the content of S to the address indicated by the stack pointer (TB+3 and TB+2) and increments the stack pointer by one.



After PUSH(632) has been used to write data into a stack, FIFO(633) and LIFO(634) can be used to read data from the stack.

Flags

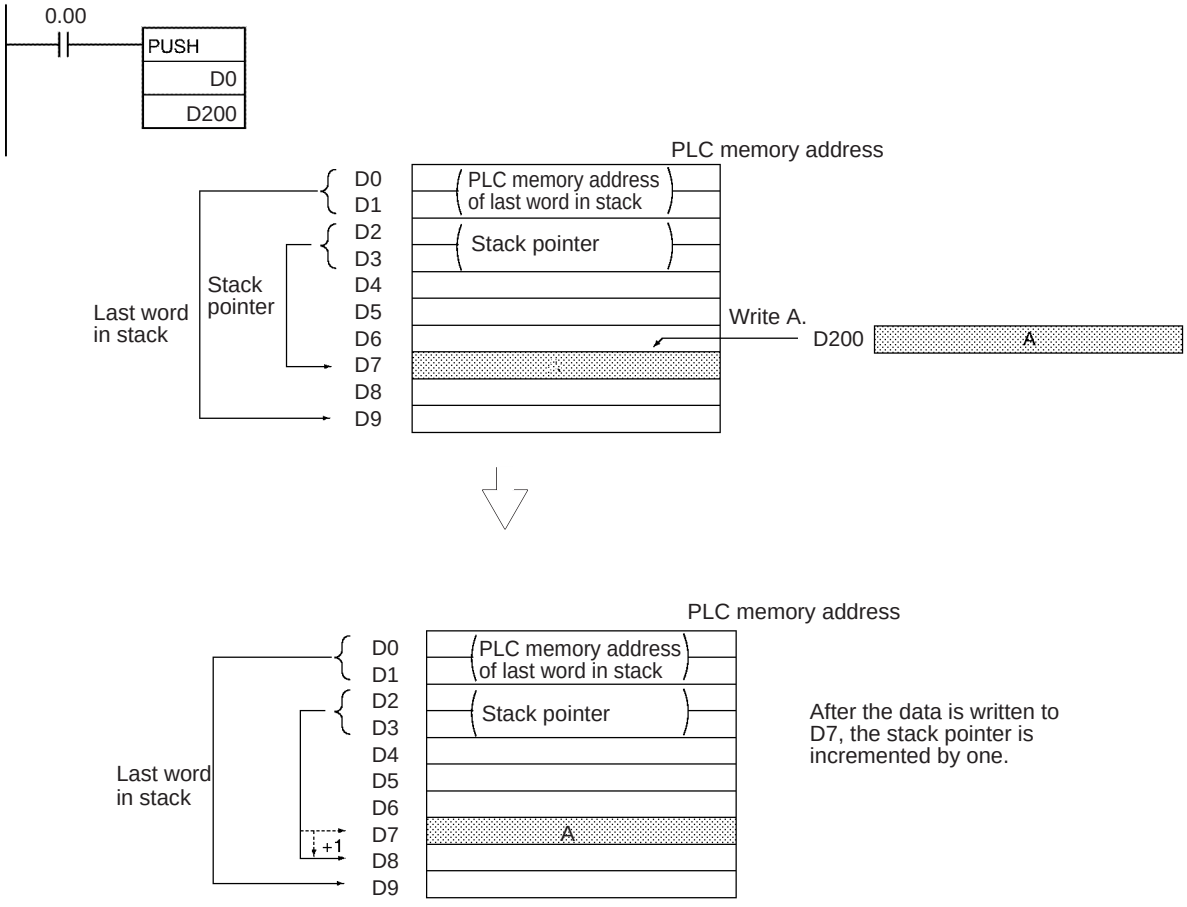
Name	Label	Operation
Error Flag	ER	ON if the address specified by the stack pointer (TB+3 and TB+2) exceeds the last word in the stack. (This is a stack overflow error.) OFF in all other cases.

Precautions

The stack must be defined in advance with SSET(630).

Examples

When CIO 0.00 is ON in the following example, PUSH(632) copies the content of D200 to the stack beginning at D0. In this case, the stack pointer indicates D7.

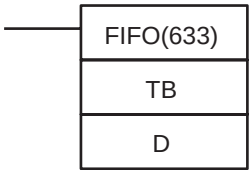


3-16-3 FIRST IN FIRST OUT: FIFO(633)

Purpose

Reads the first word of data written to the specified stack (the oldest data in the stack).

Ladder Symbol



TB: First stack address

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIFO(633)
	Executed Once for Upward Differentiation	@FIFO(633)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

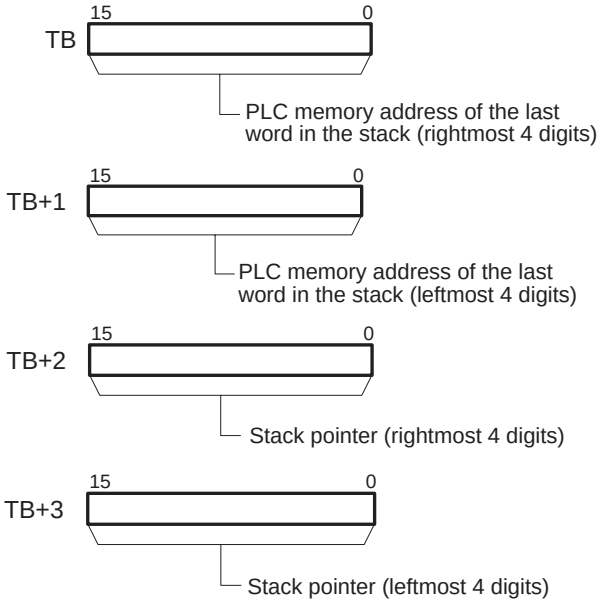
Applicable Program Areas

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

Operands

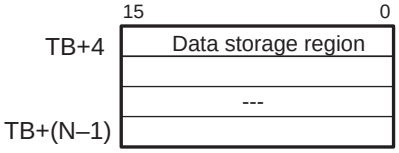
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

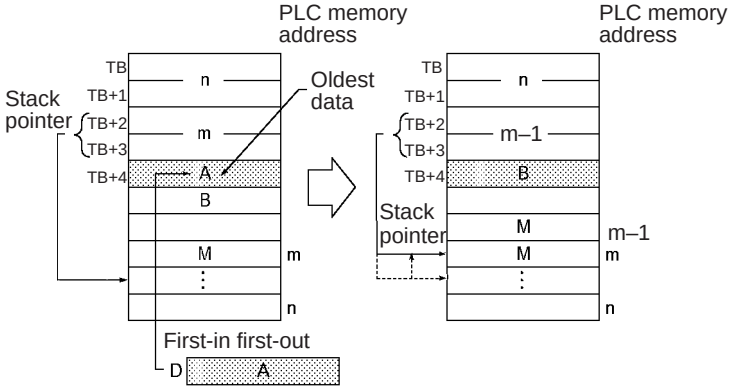


Operand Specifications

Area	TB	D
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

FIFO(633) reads the oldest word of data from the stack (TB+4) and outputs that data to D. Next, the stack pointer (TB+3 and TB+2) is decremented by one, all of the remaining data in the stack is shifted downward by one word, and the data read from TB+4 is deleted. The data at the end of the stack (the address that was indicated by the stack pointer) is left unchanged.



Use FIFO(633) in combination with PUSH(632). After PUSH(632) has been used to write data into a stack, FIFO(633) can be used to read data from the stack on a first-in first-out basis.

FIFO(633) reads the beginning data from the stack and deletes this data to move the next one forward.

Flags

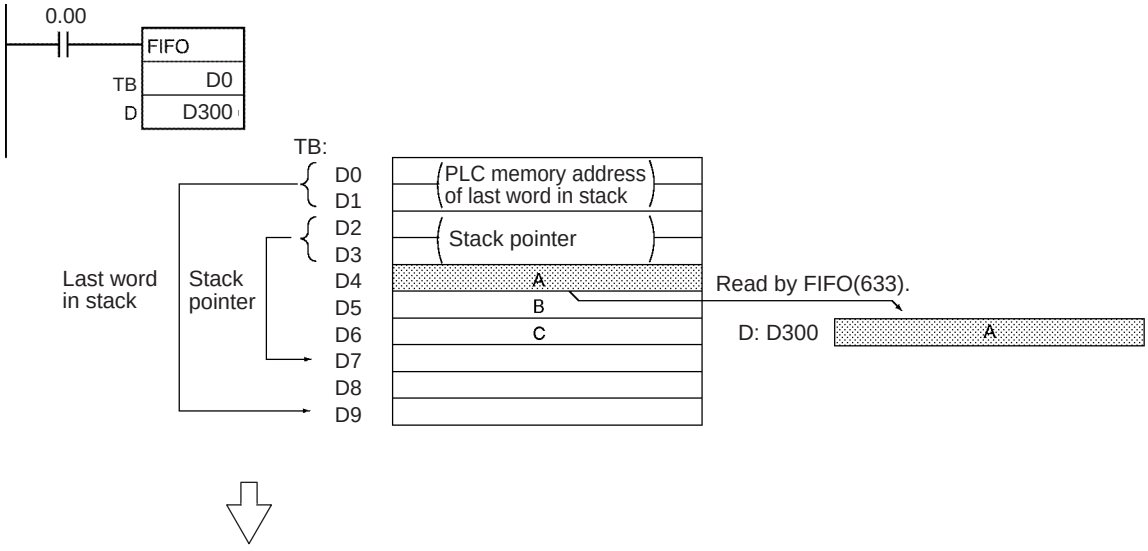
Name	Label	Operation
Error Flag	ER	ON if the contents of the stack pointer (TB+3 and TB+2) is less than or equal to the PLC memory address of first word in the data region of the stack (TB+4). (This is a stack underflow error.) OFF in all other cases.

Precautions

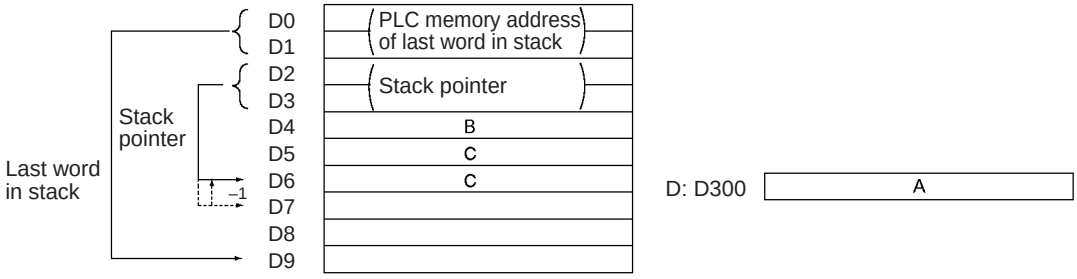
The stack must be defined in advance with SSET(630).

Examples

When CIO 0.00 is ON in the following example, FIFO(633) reads the content of D4 (TB+4 for the stack beginning at D0) and writes that data to D300.



After the data is written to D300, the stack pointer is decremented by one and the remaining data is shifted down. (The content of D5 is shifted to D4 and the content of D6 is shifted to D5.)

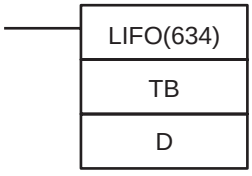


3-16-4 LAST IN FIRST OUT: LIFO(634)

Purpose

Reads the last word of data written to the specified stack (the newest data in the stack).

Ladder Symbol



TB: First stack address

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	LIFO(634)
	Executed Once for Upward Differentiation	@LIFO(634)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

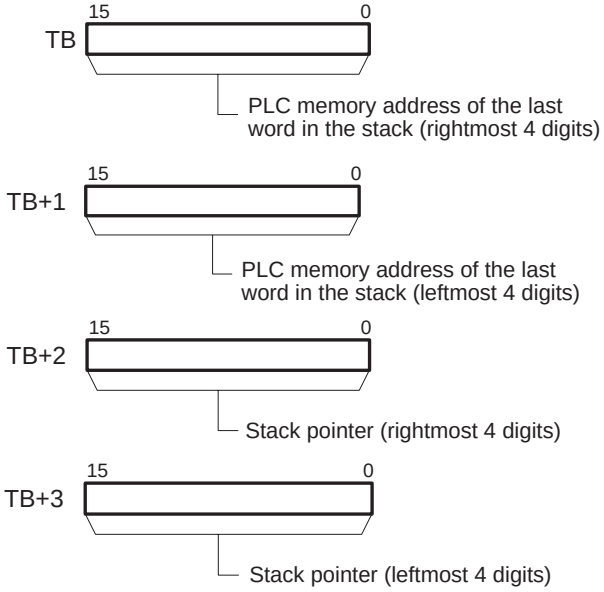
Applicable Program Areas

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

Operands

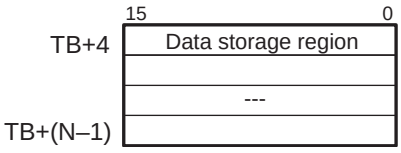
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

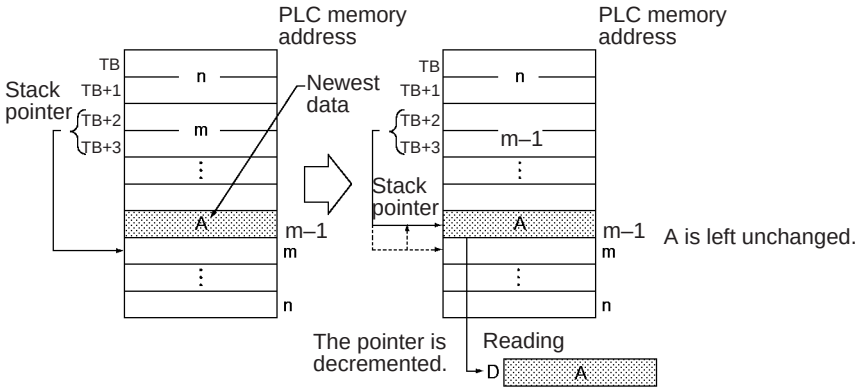


Operand Specifications

Area	TB	D
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

LIFO(634) reads the data from the address indicated by the stack pointer (the newest word of data in the stack), decrements the stack pointer by one, and outputs the data to D. The word that was read is left unchanged.



Use LIFO(634) in combination with PUSH(632). After PUSH(632) has been used to write data into a stack, LIFO(634) can be used to read data from the stack on a last-in first-out basis. After data is stored by PUSH(632), the stack pointer indicates the address next to the last data.

Flags

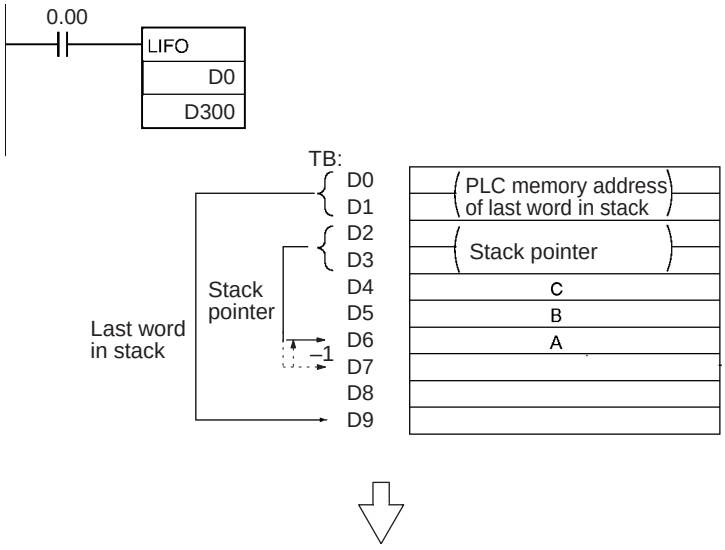
Name	Label	Operation
Error Flag	ER	ON if the contents of the stack pointer (TB+3 and TB+2) is less than or equal to the PLC memory address of first word in the data region of the stack (TB+4). (This is a stack underflow error.) OFF in all other cases.

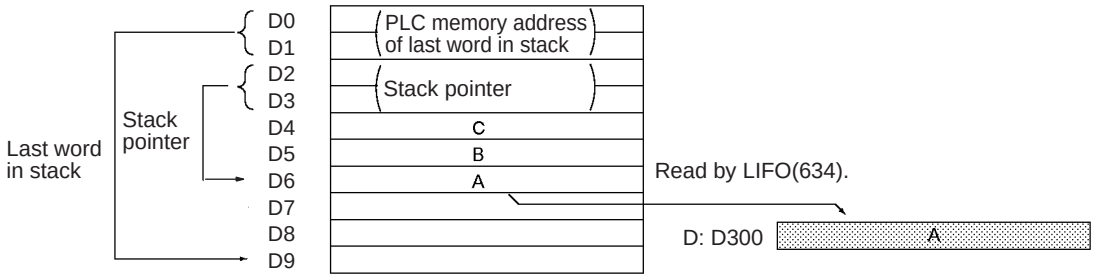
Precautions

The stack must be defined in advance with SSET(630).

Examples

When CIO 0.00 is ON in the following example, LIFO(634) reads the content of the word indicated by the stack pointer (D6) and writes that data to D300.





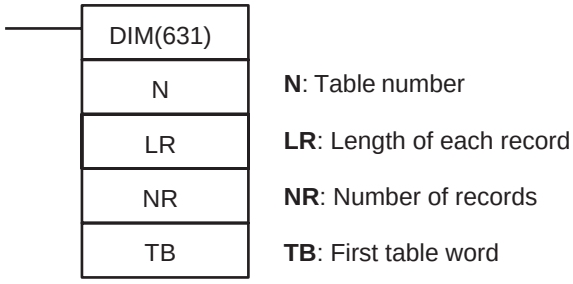
After the data is written to D300, the stack pointer is decremented by one. The content of D6 is left unchanged.

3-16-5 DIMENSION RECORD TABLE: DIM(631)

Purpose

Defines the specified I/O memory area as a record table by declaring the length of each record and the number of records. Up to 16 record tables can be defined.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	DIM(631)
	Executed Once for Upward Differentiation	@DIM(631)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Table number

Indicates the table number. N must be between 0 and15.

LR: Length of each record

Indicates the number of words in each record. LR must be 0001 to FFFF hexadecimal (1 to 65,535 words).

NR: Number of records

Indicates the number of records in the table. NR must be 0001 to FFFF hexadecimal (1 to 65,535 words).

TB: First table word

Indicates the first word of the table. All of the words in the table must be in the same data area. In other words TB and TB+LR×NR–1 must be in the same data area.

Operand Specifications

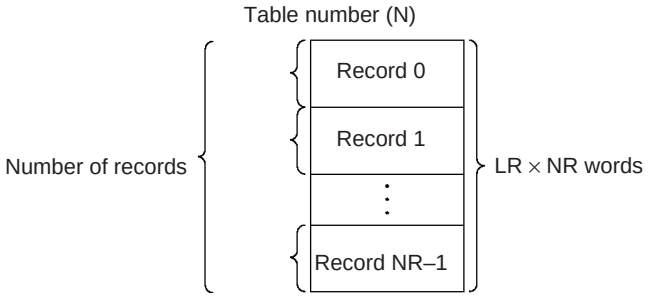
Area	N	LR	NR	TB
CIO Area	---	CIO 0 to CIO 6143		
Work Area	---	W0 to W511		
Holding Bit Area	---	H0 to H511		
Auxiliary Bit Area	---	A0 to A959		A448 to A959
Timer Area	---	T0000 to T4095		
Counter Area	---	C0000 to C4095		
DM Area	---	D0 to D32767		
Indirect DM addresses in binary	---	@ D0 to @ D32767		
Indirect DM addresses in BCD	---	*D0 to *D32767		
Constants	0 to 15	#0001 to #FFFF (binary) or &1 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

DIM(631) registers the words from TB to TB+LR×NR–1 as table number N. Table number N has NR records and each record is LR words long. The data within this region cannot be changed once the region has been declared as records.

Use DIM(631) in combination with SETR(635) (SET RECORD NUMBER) or GETR(636) (GET RECORD NUMBER) to simplify the calculation of addresses in data tables. Use DIM(631) to divide data into records and then use SETR(635) to store the first address of the desired record in an Index Register. The Index Register can then be used as a pointer in other instructions, such as read, write, search, or compare instructions.

As an example, if temperatures, pressures, or other set values are stored as records and the records for various models are combined into a table, it is easy to read the set values for each models for any particular conditions.



The two record-table instructions associated with DIM(631) are SETR(635) and GETR(636). SETR(635) sets the leading PLC memory address of the specified record number in the specified Index Register. GETR(636) outputs the record number of the record that includes the specified Index Register value (PLC memory address).

Flags

Name	Label	Operation
Error Flag	ER	ON if LR or NR is 0000. OFF in all other cases.

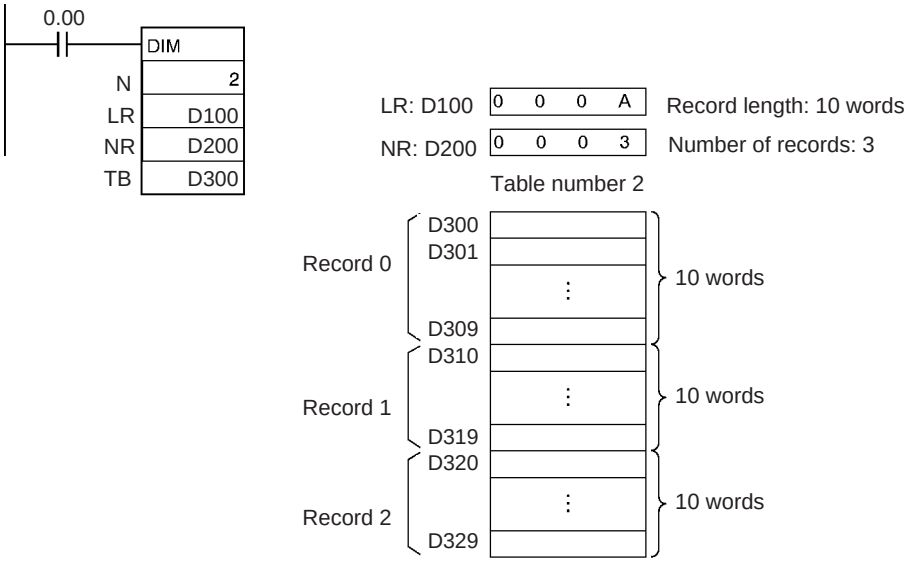
Precautions

Records in a registered table are identified by their record numbers, which range from 0 to NR–1.

Depending on the settings for the record length (LR) and number of records (NR), it is possible that a single table (from TB and TB+LR×NR–1) will overlap two data areas. Verify that no problems will arise before specifying a table that overlaps a data area boundary.

Examples

When CIO 0.00 is ON in the following example, DIM(631) defines record table number 2 with three 10-word records. The table begins at D300.

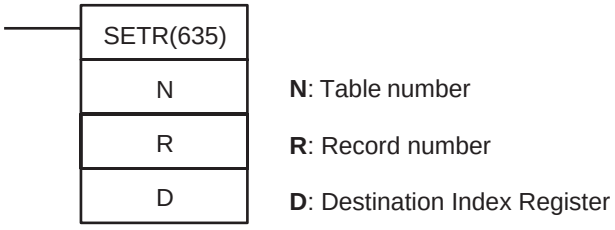


3-16-6 SET RECORD LOCATION: SETR(635)

Purpose

Writes the location of the specified record (the PLC memory address of the beginning of the record) in the specified Index Register.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SETR(635)
	Executed Once for Upward Differentiation	@SETR(635)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Table number

Indicates the table number. N must be between 0 and 15.

R: Record number

Indicates the record number of the desired record. R must be 0000 to FFFE hexadecimal (0 to 65,534). Record numbers begin with 0, so the valid record numbers are 0 to NR–1 for a table with NR records.

D: Destination Index Register

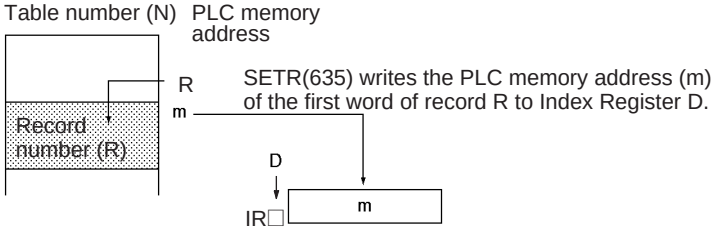
Indicates the desired Index Register. D must be IR0 to IR15.

Operand Specifications

Area	N	R	D
CIO Area	---	CIO 0 to CIO 6143	---
Work Area	---	W0 to W511	---
Holding Bit Area	---	H0 to H511	---
Auxiliary Bit Area	---	A0 to A959	---
Timer Area	---	T0000 to T4095	---
Counter Area	---	C0000 to C4095	---
DM Area	---	D0 to D32767	---
Indirect DM addresses in binary	---	@ D0 to @ D32767	---
Indirect DM addresses in BCD	---	*D0 to *D32767	---
Constants	0 to 15	#0000 to #FFFE (binary) or &0 to 65534	---
Data Registers	---	DR0 to DR15	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,– (– –)IR0 to, – (– –)IR15	---

Description

SETR(635) stores the PLC memory address of the first word of the specified record in the specified Index Register. The following diagram shows the basic operation of SETR(635).



Flags

Name	Label	Operation
Error Flag	ER	ON if the specified table number (N) has not been defined with DIM(631). ON if the specified record number (R) exceeds the highest record number in the table (NR–1). OFF in all other cases.

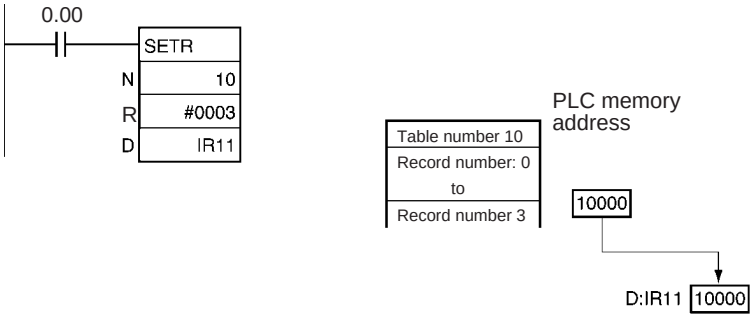
Precautions

The record table must be defined in advance with DIM(631).

Valid record numbers range from 0 to NR–1, where NR is the number of records specified when the table was defined with DIM(631).

Examples

When CIO 0.00 is ON in the following example, SETR(635) finds the PLC memory address of the first word of record 3 of table number 10 and stores this address in Index Register IR11.

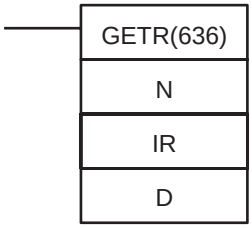


3-16-7 GET RECORD NUMBER: GETR(636)

Purpose

Returns the record number of the record at the PLC memory address contained in the specified Index Register.

Ladder Symbol



N: Table number

IR: Index Register

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	GETR(636)
	Executed Once for Upward Differentiation	@GETR(636)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Table number

Indicates the table number. N must be between 0 and 15.

IR: Index Register

Indicates the desired Index Register. D must be IR0 to IR15.

D: Destination word

Indicates the word where the record number will be written.

Operand Specifications

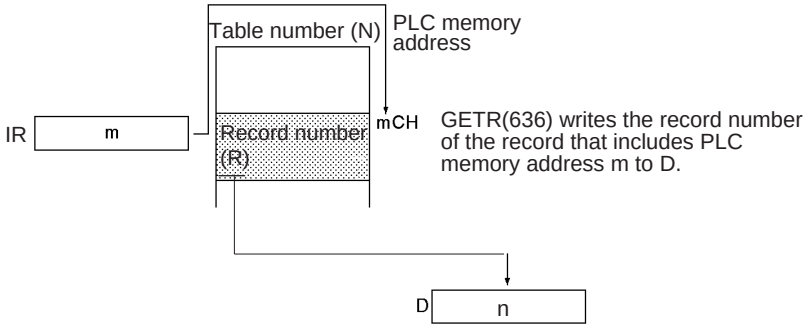
Area	N	IR	D
CIO Area	---		CIO 0 to CIO 6143
Work Area	---		W0 to W511
Holding Bit Area	---		H0 to H511
Auxiliary Bit Area	---		A448 to A959

Area	N	IR	D
Timer Area	---		T0000 to T4095
Counter Area	---		C0000 to C4095
DM Area	---		D0 to D32767
Indirect DM addresses in binary	---		@ D0 to @ D32767
Indirect DM addresses in BCD	---		*D0 to *D32767
Constants	0 to 15	---	---
Data Registers	---		DR0 to DR15
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

GETR(636) finds which record includes the PLC memory address contained in the specified Index Register and writes that record number in D. The PLC memory address contained in the Index Register does not have to be the first word in the record; it can be any word in the record.

The following diagram shows the basic operation of GETR(636).



Flags

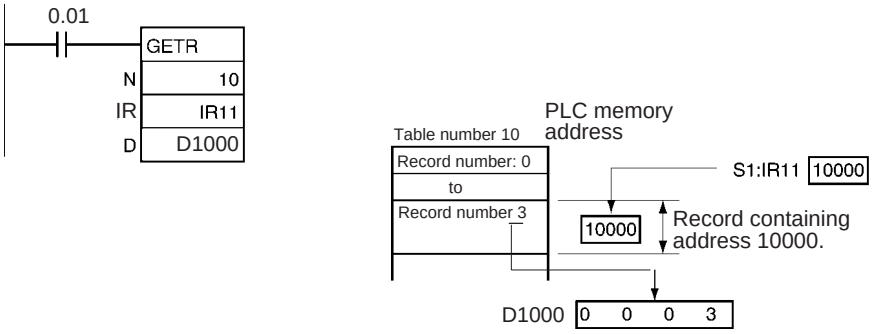
Name	Label	Operation
Error Flag	ER	ON if the PLC memory address in the specified Index Register is not within the specified table (N). ON if the specified table number (N) has not been defined with DIM(631). OFF in all other cases.

Precautions

The record table must be defined in advance with DIM(631) and the PLC memory address in the specified Index Register must be within the specified table.

Examples

When CIO 0.01 is ON in the following example, GETR(636) finds the record number of the record that contains the PLC memory address in Index Register IR11 and writes this record number to D1000.

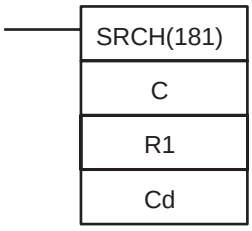


3-16-8 DATA SEARCH: SRCH(181)

Purpose

Searches for a word of data within a range of words.

Ladder Symbol



C: First control word

R1: First word in range

Cd: Comparison data

Variations

Variations		
	Executed Each Cycle for ON Condition	SRCH(181)
	Executed Once for Upward Differentiation	@SRCH(181)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

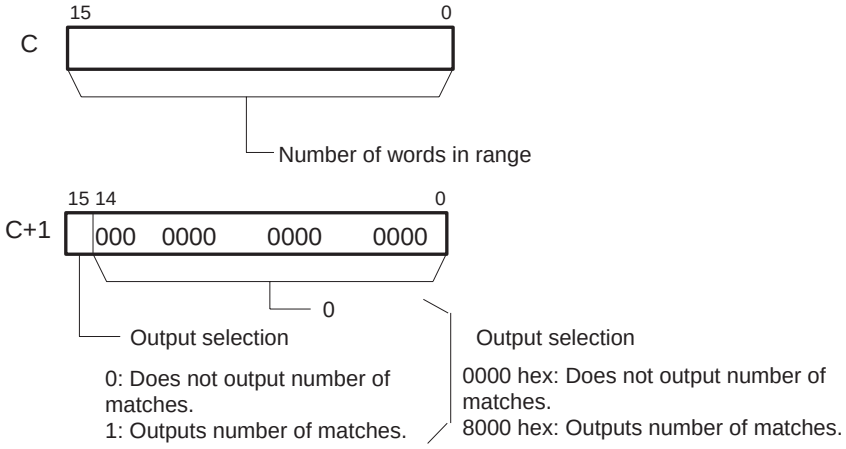
Applicable Program Areas

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

Operands

C and C+1: Control words

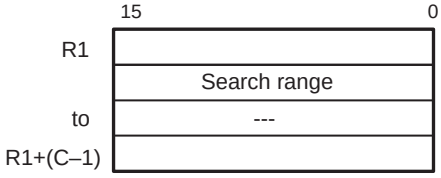
C specifies the number of words in the range and bit 15 of C+1 indicates whether or not to output the number of matches to DR0.



Note C and C+1 must be in the same data area.

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 desired data. (C is the number of words set in C.)



Note R1 and R1+C–1 must be in the same data area.

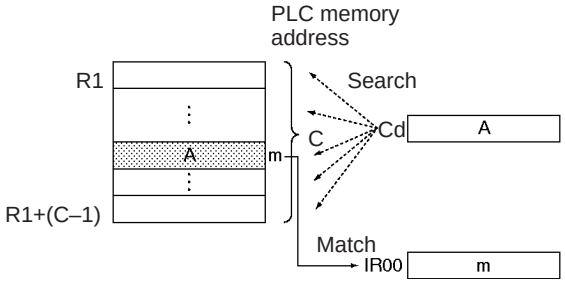
Operand Specifications

Area	C	R1	Cd
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143	
Work Area	W0 to W510	W0 to W511	
Holding Bit Area	H0 to H510	H0 to H511	
Auxiliary Bit Area	A0 to A958	A0 to A959	
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D0 to D32766	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	Specified values only	---	#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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SRCH(181) searches the range of memory from R1 to R1+C–1 for words that contain the comparison data (Cd). If a match is found, SRCH(181) writes the PLC memory address of the word to IR0 and turns the Equals Flag ON. (If there are two or more matches, just the address of the first word containing the comparison data is written to IR0.)

When bit 15 of C+1 has been set to 1, SRCH(181) writes the number of matches to DR0. When bit 15 of C+1 is 0, DR0 is left unchanged.



SRCH(181) searches table data that contains one word in each record. For searching data that contains more than one word per record, use DIM(631), SETR(635), GETR(636), FOR(512)–NEXT(513), or BREAK(514) together with an Index Register (IR).

The status of the Equals Flag can be checked immediately after execution to determine whether or not there was a match.

Flags

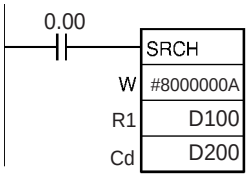
Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if one or more of the words in the search range contain the comparison data. OFF in all other cases.

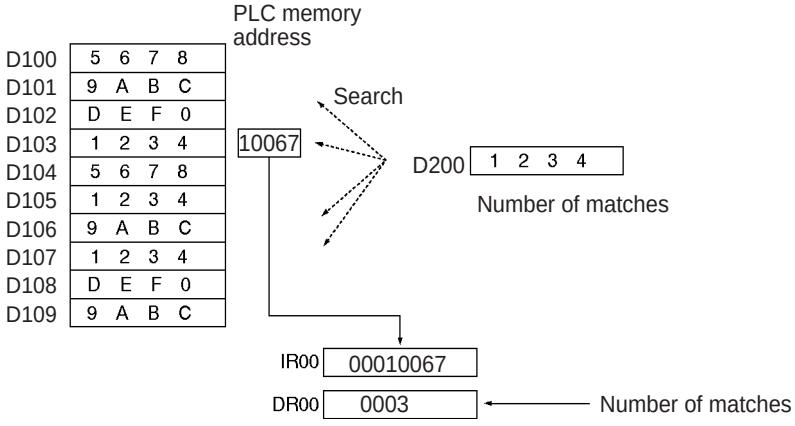
Precautions

If no match is found, the contents of IR0 and DR0 are left unchanged.

Examples

When CIO 0.00 is ON in the following example, SRCH(181) searches the 10-word range beginning at D100 for words that have the same content as D200. The PLC memory address of the first word containing a match is written to IR0 and the total number of matches is written to DR0.





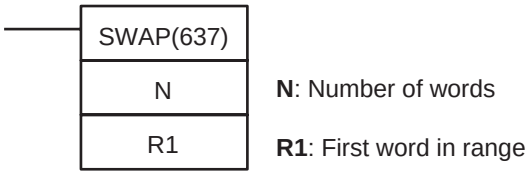
If the table length is specified as &10 (10 decimal) or A hexadecimal, the number of matches will not be output to the data register DR0.

3-16-9 SWAP BYTES: SWAP(637)

Purpose

Switches the leftmost and rightmost bytes in all of the words in the range.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SWAP(637)
	Executed Once for Upward Differentiation	@SWAP(637)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

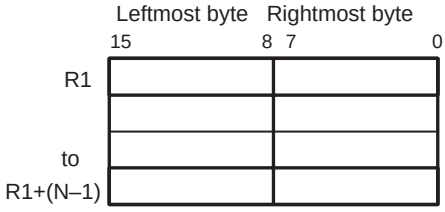
Operands

N: Number of words

N specifies the number of words in the range and must be 0001 to FFFF hexadecimal (or &1 to &65,535).

R1: First word in range

R1 specifies the first word in the range.

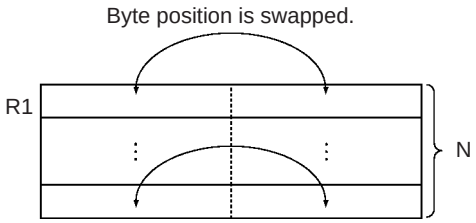


Operand Specifications

Area	N	R1
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0001 to #FFFF (binary) or &1 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SWAP(637) switches the position of the two bytes in all of the words in the range of memory from R1 to R1+N–1. This instruction can be used to reverse the order of ASCII-code characters in each word.

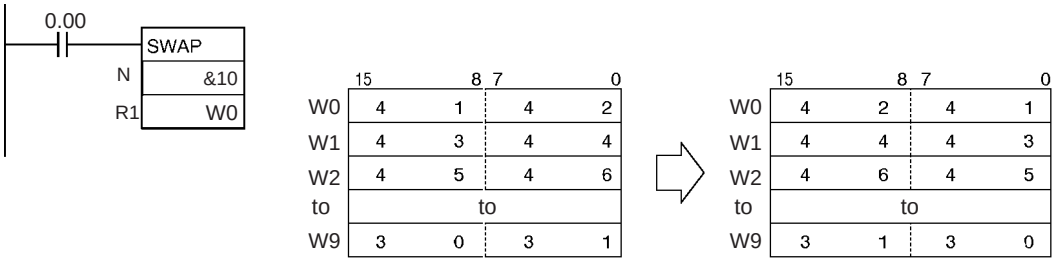


Flags

Name	Label	Operation
Error Flag	ER	ON if the N is 0000. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Examples

When CIO 0.00 is ON in the following example, SWAP(637) switches the data in the leftmost bytes with the data in the rightmost bytes in each word in the 10-word range from W0 to W9.

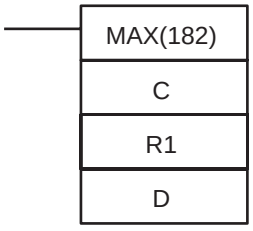


3-16-10 FIND MAXIMUM: MAX(182)

Purpose

Finds the maximum value in the 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.
Immediate Refreshing Specification		Not supported.

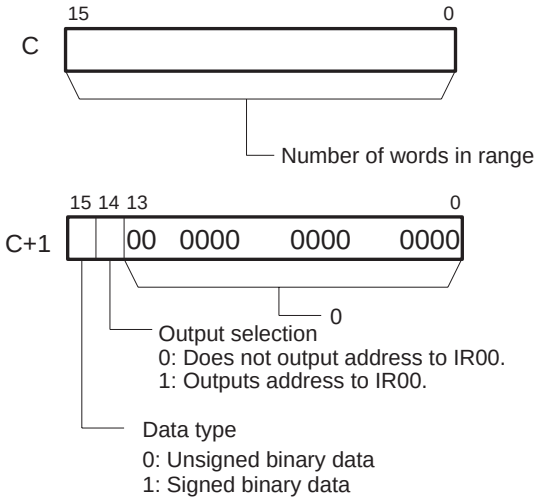
Applicable Program Areas

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 PLC memory address of the word that contains the maximum value to IR0.

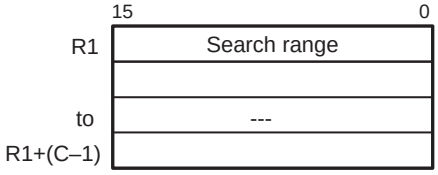


The following table shows the possible values of C.

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.)



Operand Specifications

Area	C	R1	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143	
Work Area	W0 to W510	W0 to W511	
Holding Bit Area	H0 to H510	H0 to H511	
Auxiliary Bit Area	A0 to A958	A0 to A959	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D0 to D32766	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	Specified values only	---	

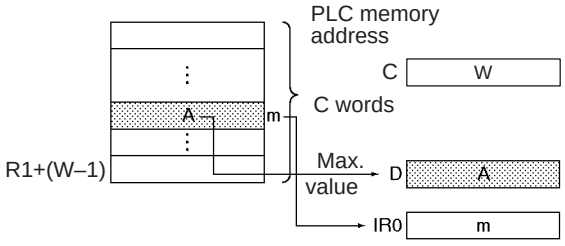
Area	C	R1	D
Data Registers	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

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 PLC memory address of the word containing the maximum value to IR0. (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 IR0.)

When bit 15 of C+1 has been set to 1, MAX(182) treats the data within the range as signed binary data.



Flags

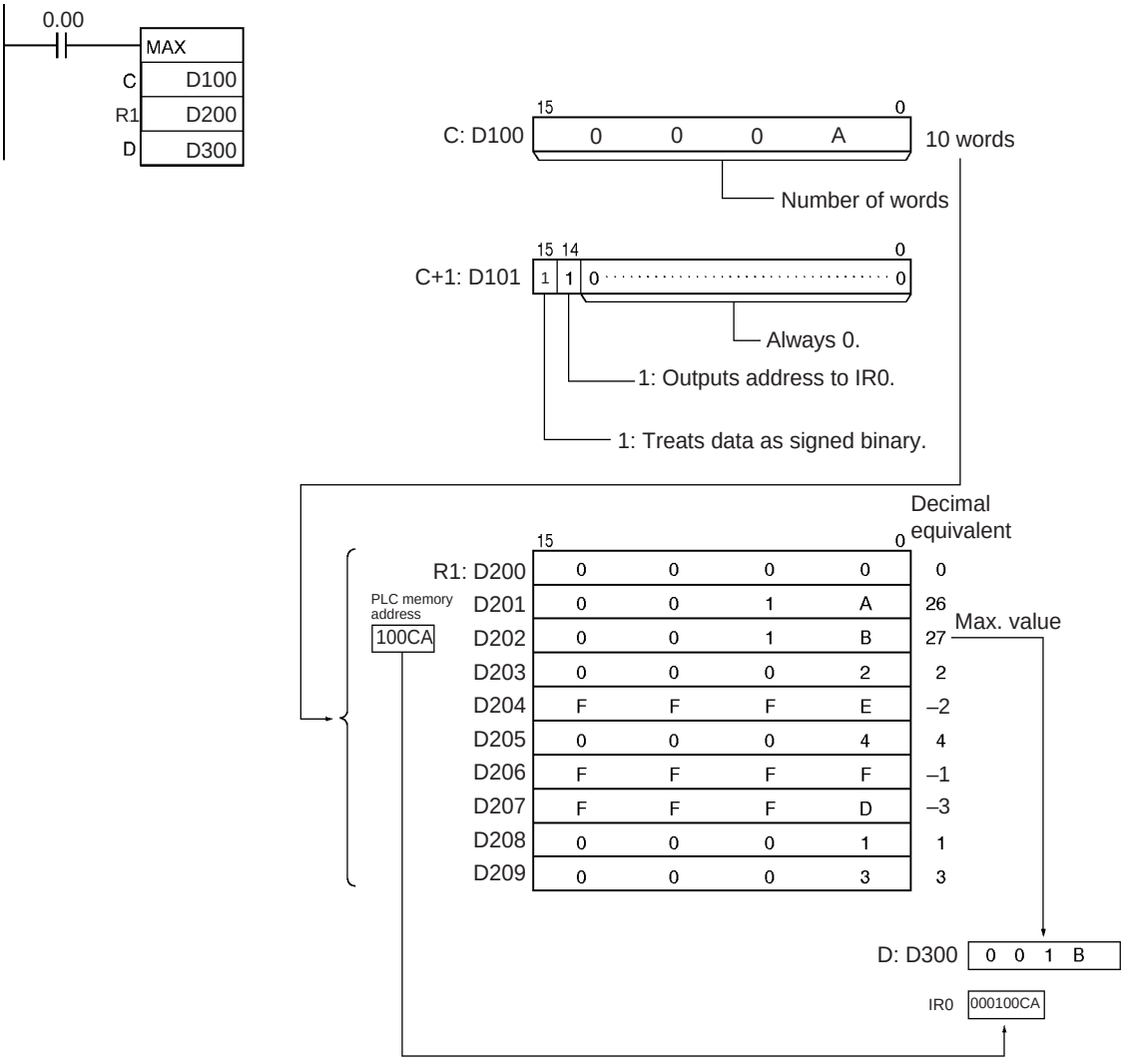
Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. 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 0.00 turns ON in the following example, MAX(182) searches the 10-word range (specified in D100) beginning at D200 for the maximum value. The maximum value is written to D300 and the PLC memory address of the word containing the maximum value is written to IR0.

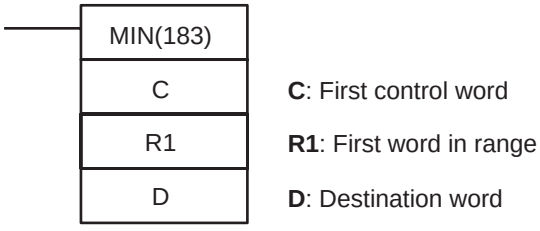


3-16-11 FIND MINIMUM: MIN(183)

Purpose

Finds the minimum value in the range.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MIN(183)
	Executed Once for Upward Differentiation	@MIN(183)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

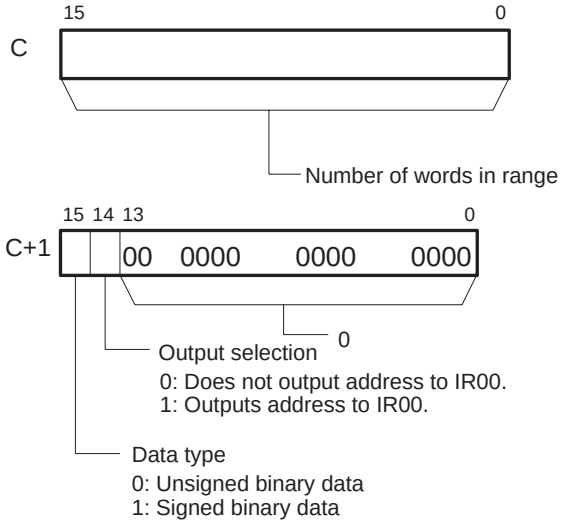
Applicable Program Areas

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 PLC memory address of the word that contains the minimum value to IR0.

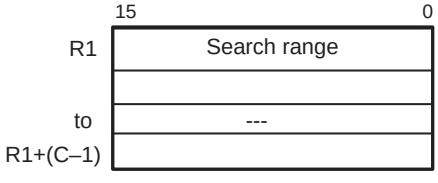


The following table shows the possible values of C.

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.)



Operand Specifications

Area	C	R1	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143	
Work Area	W0 to W510	W0 to W511	
Holding Bit Area	H0 to H510	H0 to H511	
Auxiliary Bit Area	A0 to A958	A0 to A959	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	

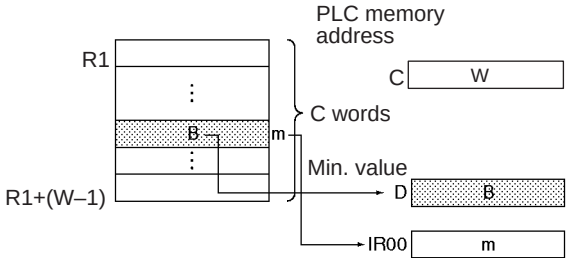
Area	C	R1	D
DM Area	D0 to D32766	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	Specified values only	---	
Data Registers	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

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 PLC memory address of the word containing the minimum value to IR0. (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 IR0.)

When bit 15 of C+1 has been set to 1, MIN(183) treats the data within the range as signed binary data.



Flags

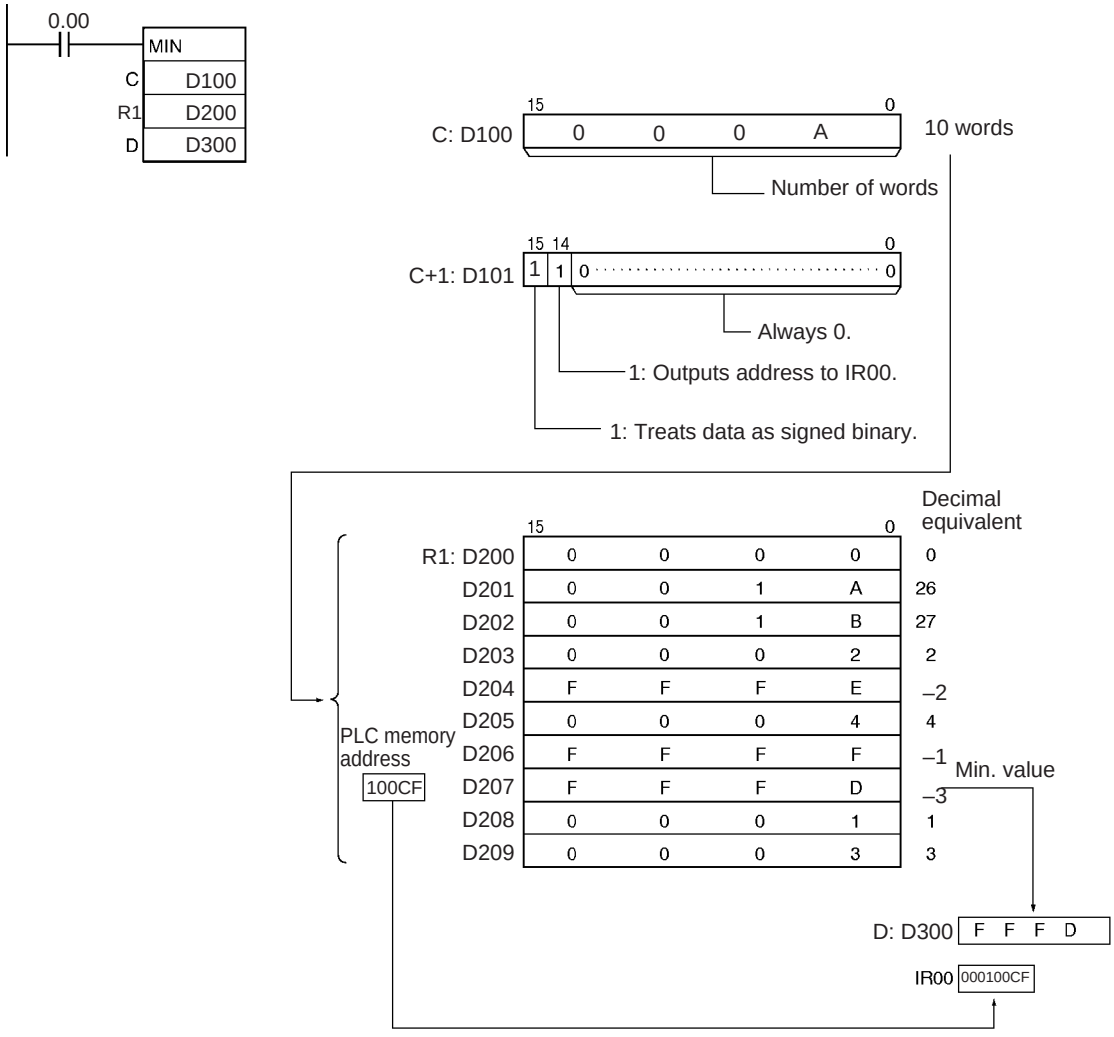
Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. 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 0.00 turns ON in the following example, MIN(183) searches the 10-word range (specified in D100) beginning at D200 for the minimum value. The minimum value is written to D300 and the PLC memory address of the word containing the minimum value is written to IR0.

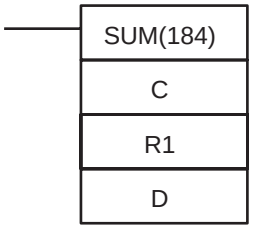


3-16-12 SUM: SUM(184)

Purpose

Adds the bytes or words in the range and outputs the result to two words.

Ladder Symbol



C: First control word

R1: First word in range

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SUM(184)
	Executed Once for Upward Differentiation	@SUM(184)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

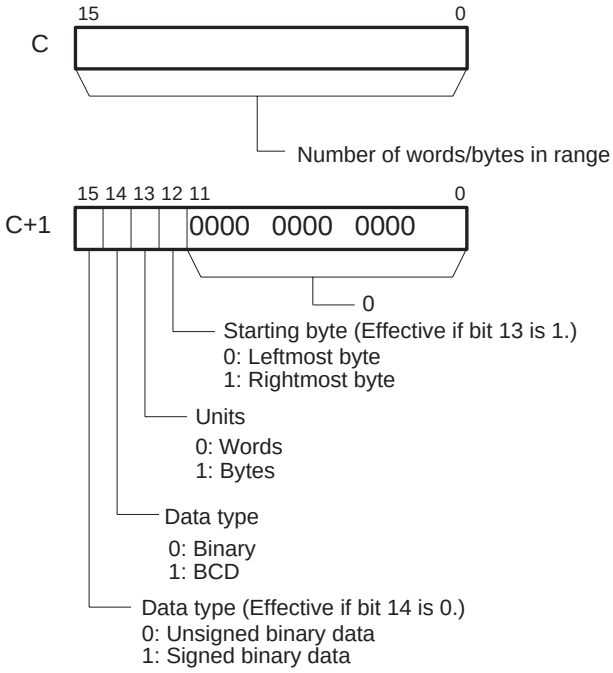
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

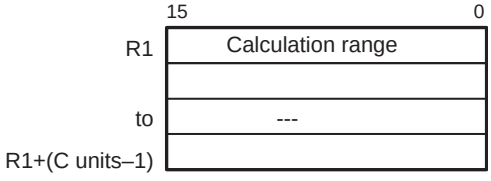
C specifies the number of units (bytes or words) to be summed. (Bit 13 of C+1 determines whether bytes or words are being summed.)

Bits 12 to 15 of C+1 indicate what type of data is being summed, as shown in the following diagram.



R1: First word in range

R1 specifies the first word in the range. The length of the range depends on the number of units as well as the starting byte, if bytes are being added.



D: First destination word

The result of the calculation is output to D+1 and D. The leftmost four digits are stored in D+1 and the rightmost four digits are stored in D.

Operand Specifications

Area	C	R1	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143	CIO 0 to CIO 6142
Work Area	W0 to W510	W0 to W511	W0 to W510
Holding Bit Area	H0 to H510	H0 to H511	H0 to H510
Auxiliary Bit Area	A0 to A958	A0 to A959	A448 to A958
Timer Area	T0000 to T4094	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4094	C0000 to C4095	C0000 to C4094

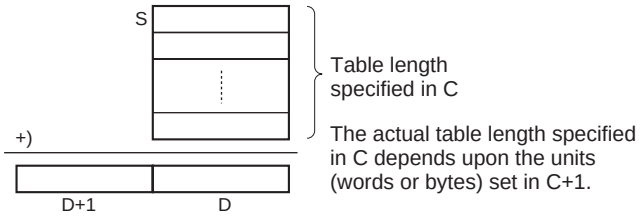
Area	C	R1	D
DM Area	D0 to D32766	D0 to D32767	D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	Specified values only	---	
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SUM(184) adds C units of data beginning with the data in R1 and outputs the result to D+1 and D. The settings in C+1 determine whether the units are words or bytes, whether the data is binary (signed or unsigned) or BCD, and whether to start with the right or left byte of R1 if bytes are being added.

When bit 14 of C+1 has been set to 0, SUM(184) treats the data as binary. In this case, bit 15 determines whether the data is signed (bit 15 = 1) or unsigned (bit 15 = 0).

When bit 13 of C+1 has been set to 1, SUM(184) adds bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).

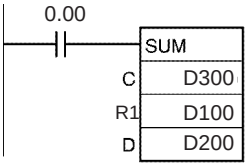


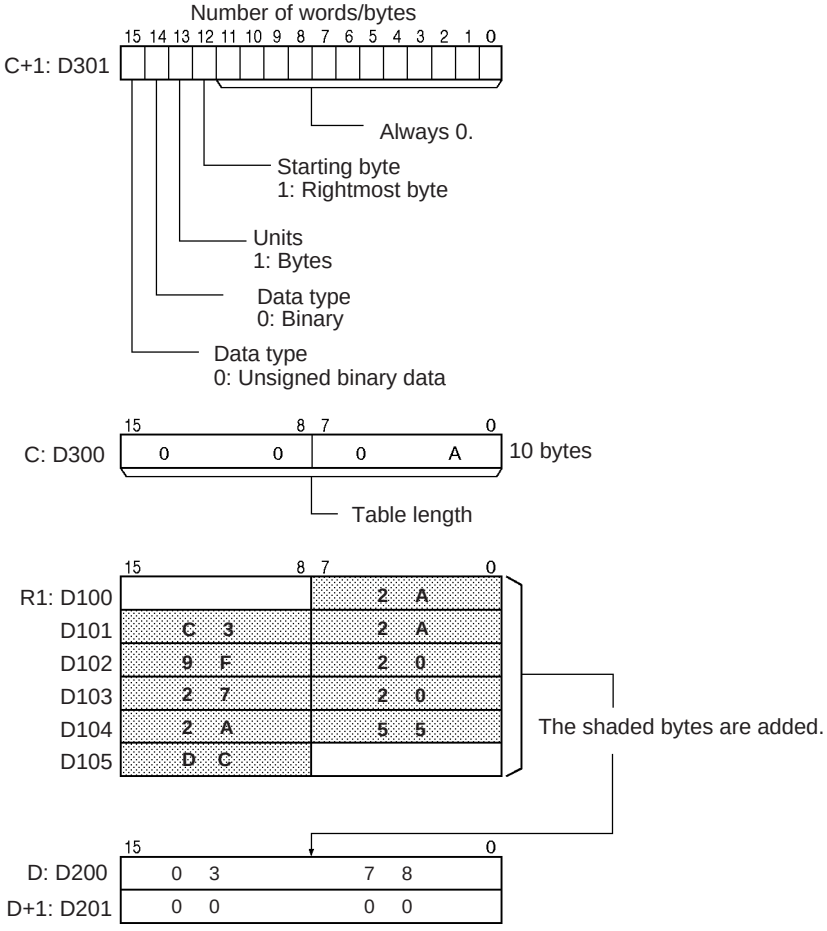
Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the BCD data has been specified, but the range contains binary data. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. 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 is ON in the result. OFF in all other cases.

Examples

When CIO 0.00 is ON in the following example, SUM(184) adds 10 bytes (specified in D300) of unsigned binary data beginning with the rightmost byte of D100 and writes the result to D201 and D200.



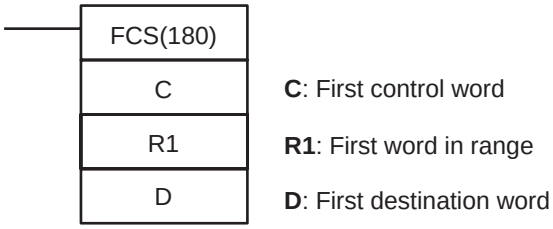


3-16-13 FRAME CHECKSUM: FCS(180)

Purpose

Calculates the FCS value for the specified range and outputs the result in ASCII.

Ladder Symbol



C: First control word

R1: First word in range

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FCS(180)
	Executed Once for Upward Differentiation	@FCS(180)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

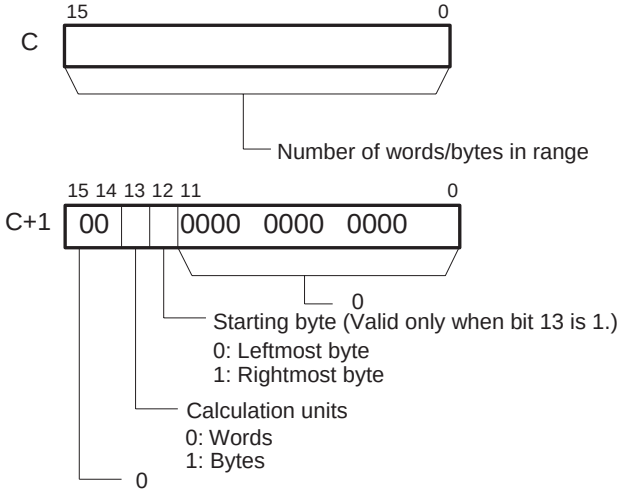
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

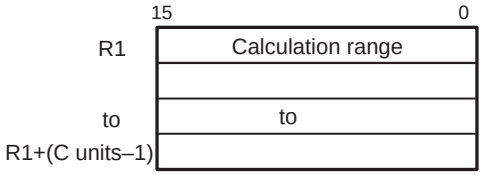
C specifies the number of units (bytes or words) to be used in the FCS calculation. (Bit 13 of C+1 determines whether bytes or words are being used.)

When bit 13 of C+1 has been set to 1, FCS(180) calculates the FCS value for bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).



R1: First word in range

R1 specifies the first word in the range. The length of the range depends on the number of units as well as the starting byte, if bytes are being used in the calculation.



D: First destination word

The result of the calculation is output to D if bytes have been selected.

The result of the calculation is output to D+1 and D if words have been selected. In this case, the leftmost four digits are stored in D+1 and the rightmost four digits are stored in D.

Operand Specifications

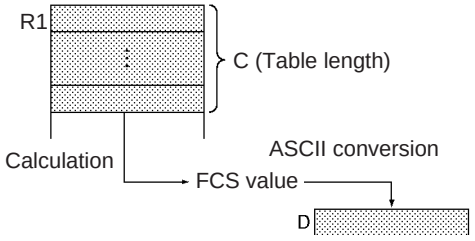
Area	C	R1	D
CIO Area	CIO 0 to CIO 6142	CIO 0 to CIO 6143	
Work Area	W0 to W510	W0 to W511	
Holding Bit Area	H0 to H510	H0 to H511	
Auxiliary Bit Area	A0 to A958	A0 to A959	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D0 to D32766	D0 to D32767	

Area	C	R1	D
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	Specified values only	---	
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

FCS(180) calculates the FCS value for C units of data beginning with the data in R1, converts the value to ASCII code, and outputs the result to D (for bytes) or D+1 and D (for words). The settings in C+1 determine whether the units are words or bytes, whether the data is binary (signed or unsigned) or BCD, and whether to start with the right or left byte of R1 if bytes are being added.

When bit 13 of C+1 has been set to 1, FCS(180) operates on bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).

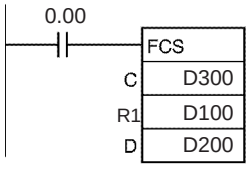


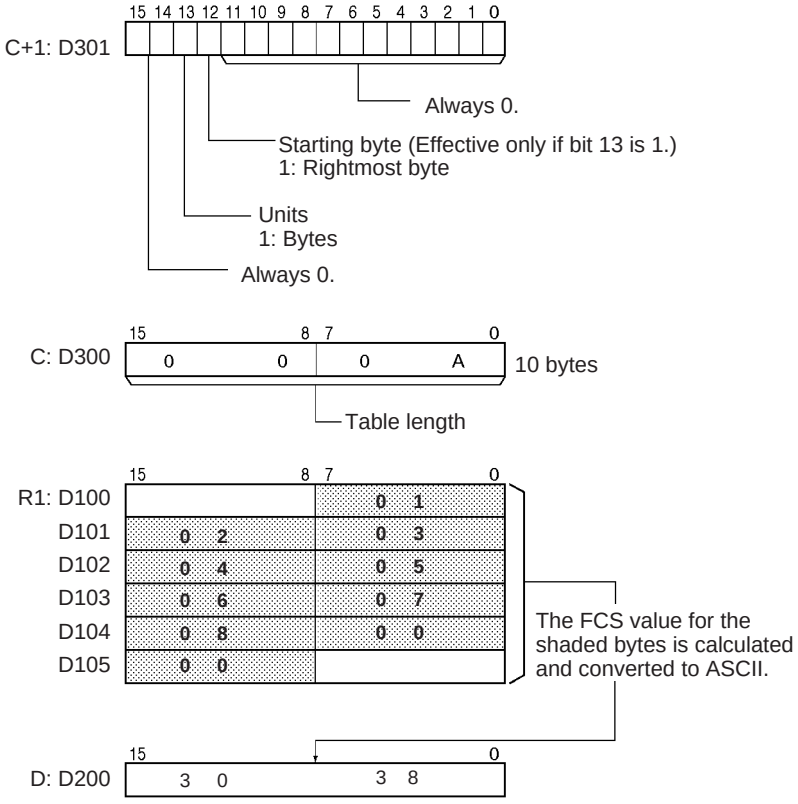
Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Examples

When CIO 0.00 is ON in the following example, FCS(180) calculates the FCS value for the 10 bytes (specified in D300) of data beginning with the rightmost byte of D100 and writes the result to D200.



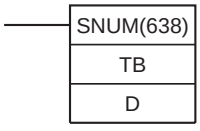


3-16-14 STACK SIZE READ: SNUM(638)

Purpose

Counts the amount of stack data (number of words) in the specified stack.

Ladder Symbol



TB: First stack address
D: Destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	SNUM(638)
	Executed Once for Upward Differentiation	@SNUM(638)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

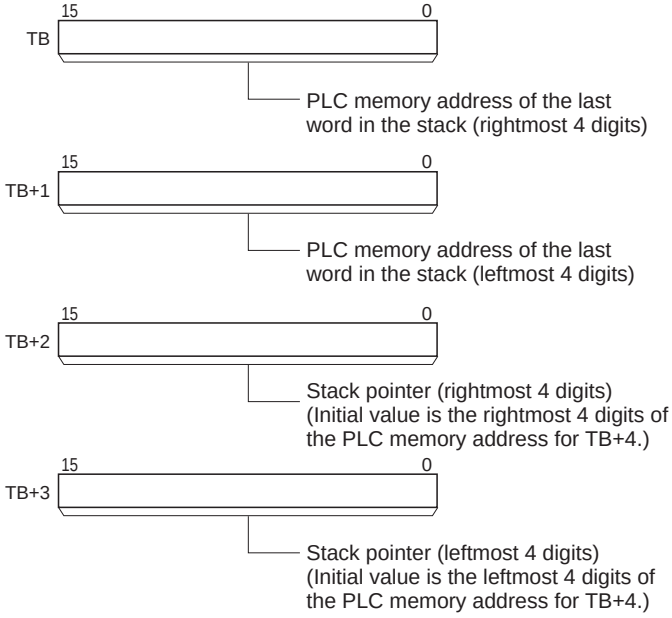
Applicable Program Areas

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

Operands

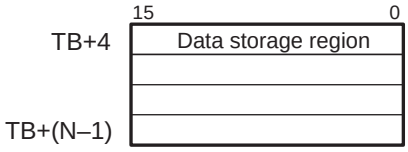
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

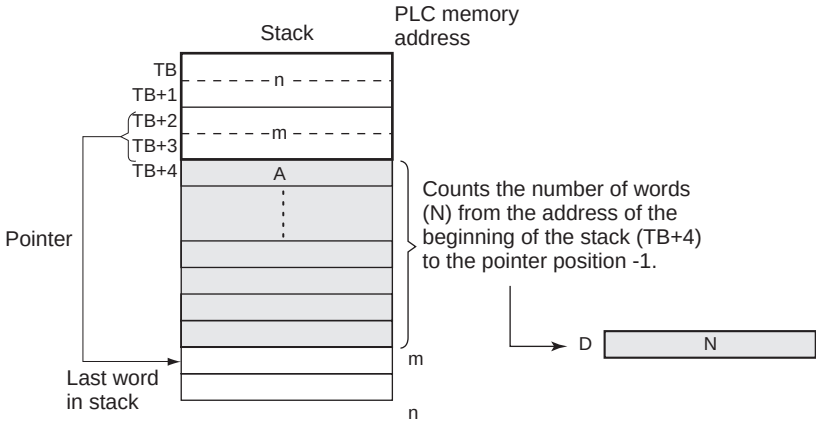


Operand Specifications

Area	TB	D
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SNUM(638) counts the number of data words in the specified stack from the beginning of the data region at TB+4 to the address before the one indicated by the stack pointer (TB+3 and TB+2). SNUM(638) does not change the data in the stack or the stack pointer.



Flags

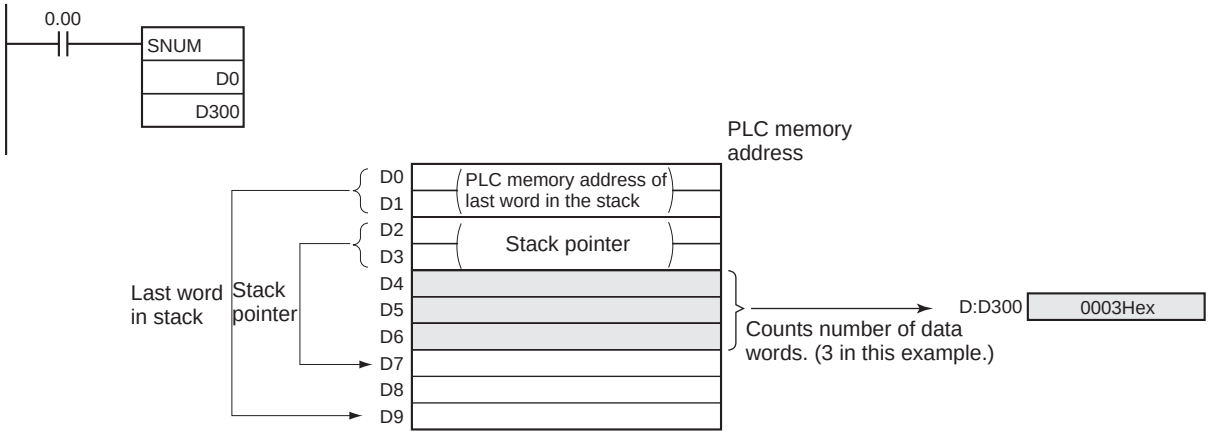
Name	Label	Operation
Error Flag	ER	ON if the number of words of data in the stack (the value output to D) is 0. OFF in all other cases.

Precautions

The stack must be defined in advance with SSET(630).

Examples

When CIO 0.00 is ON in the following example, SNUM(638) counts the number of words from the beginning of the data region at D4 to the stack pointer position - 1 (D6) and outputs the result to D300. (In this case, the stack pointer indicates D7.) The stack area begins at D0.

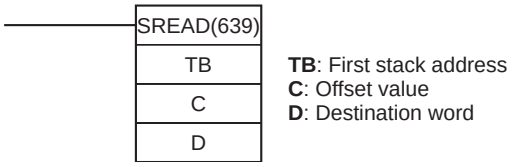


3-16-15 STACK DATA READ: SREAD(639)

Purpose

Reads the data from the specified data element in the stack. The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SREAD(639)
	Executed Once for Upward Differentiation	@SREAD(639)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

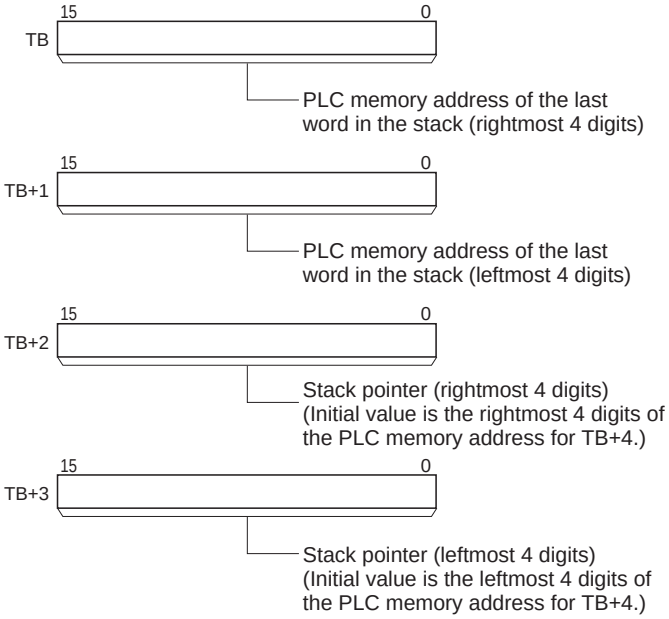
Applicable Program Areas

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

Operands

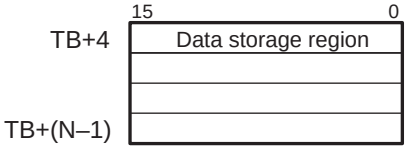
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

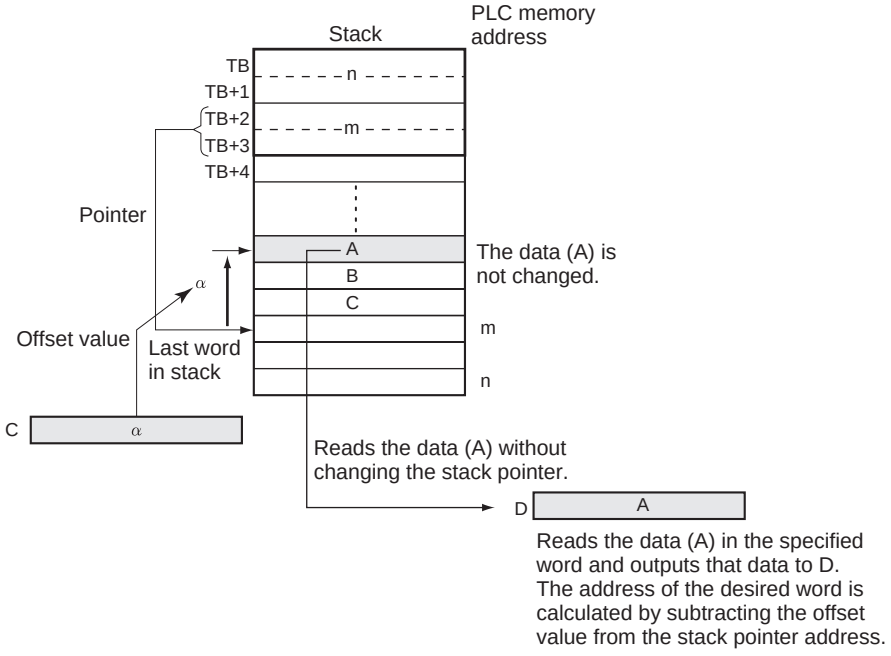


Operand Specifications

Area	TB	C	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	---		
Index Registers	---	#0001 to #FFFB (Hexadecimal)	---
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	DR0 to DR15	

Description

SREAD(639) reads the data from the address specified by the stack pointer (TB+3 and TB+2) minus the offset value in C. SREAD(639) does not change the data in the stack or the stack pointer.



SREAD(639) can be used to read the data for an item currently on a conveyor. The position of the desired item is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified read location is not within the stack area. ON if the offset value specified in C is 0 or greater than the maximum data region size (FFFB hex). OFF in all other cases.
Equals Flag	=	ON if the output data in D is 0000. OFF in all other cases.

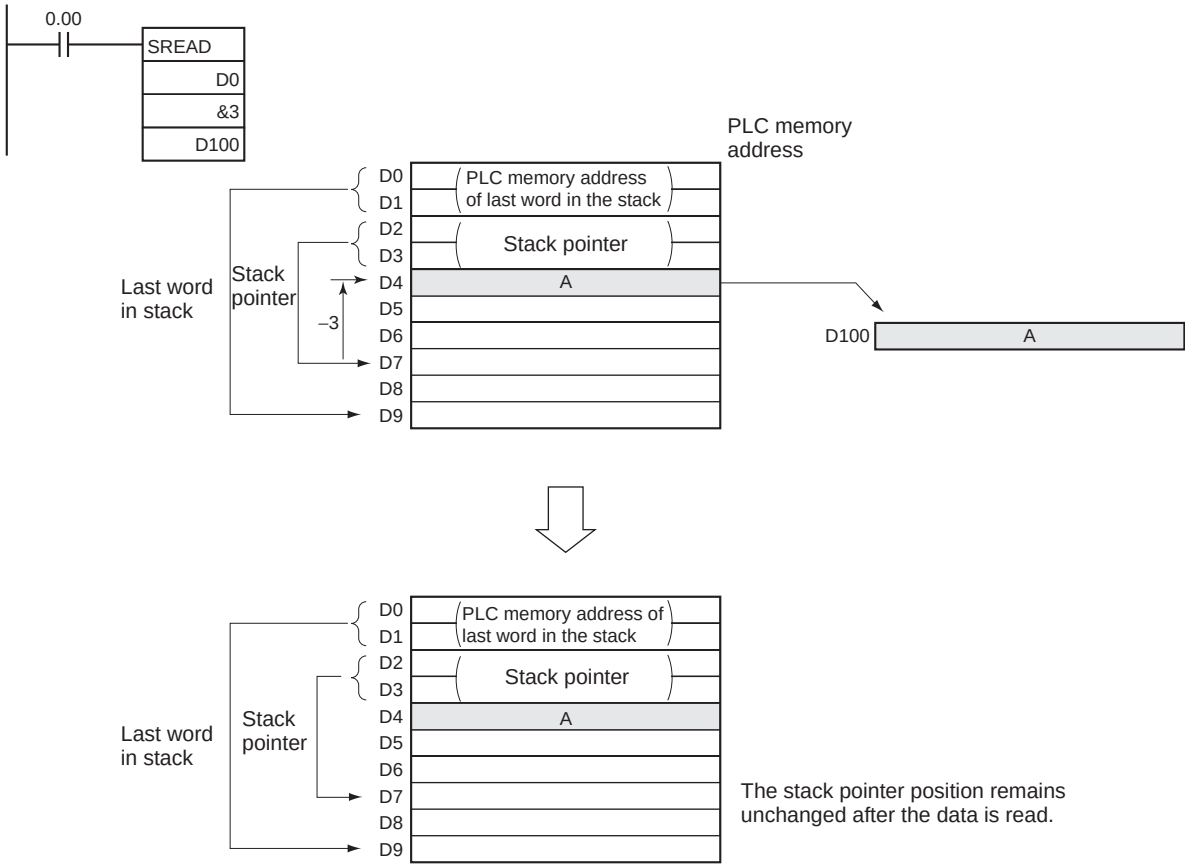
Precautions

The stack must be defined in advance with SSET(630).

The address in the stack pointer must be greater than the PLC memory address of the beginning of the data region (TB+4). An error will occur if the stack pointer is less than the PLC memory address of TB+4, i.e., if a stack underflow error occurs.

Examples

When CIO 0.00 is ON in the following example, SREAD(639) reads the data in the specified word in the stack starting at D0 and outputs the data to D100. In this case, the stack pointer indicates D7 and the offset value is 3, so the data is read from D4.

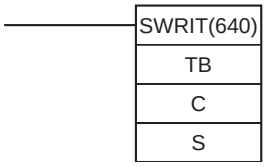


3-16-16 STACK DATA OVERWRITE: SWRIT(640)

Purpose

Writes the source data to the specified data element in the stack (overwriting the existing data). The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SWRIT(640)
	Executed Once for Upward Differentiation	@SWRIT(640)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

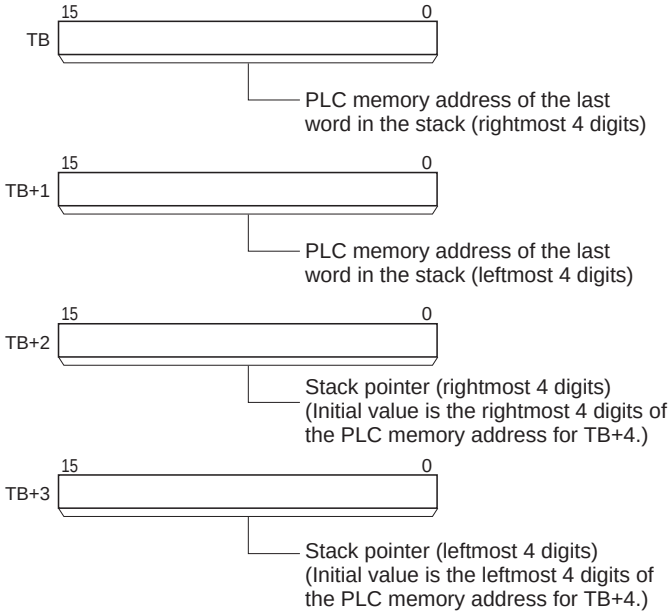
Applicable Program Areas

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

Operands

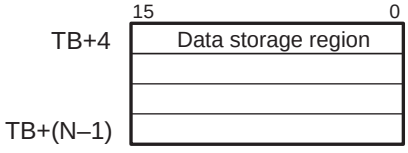
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

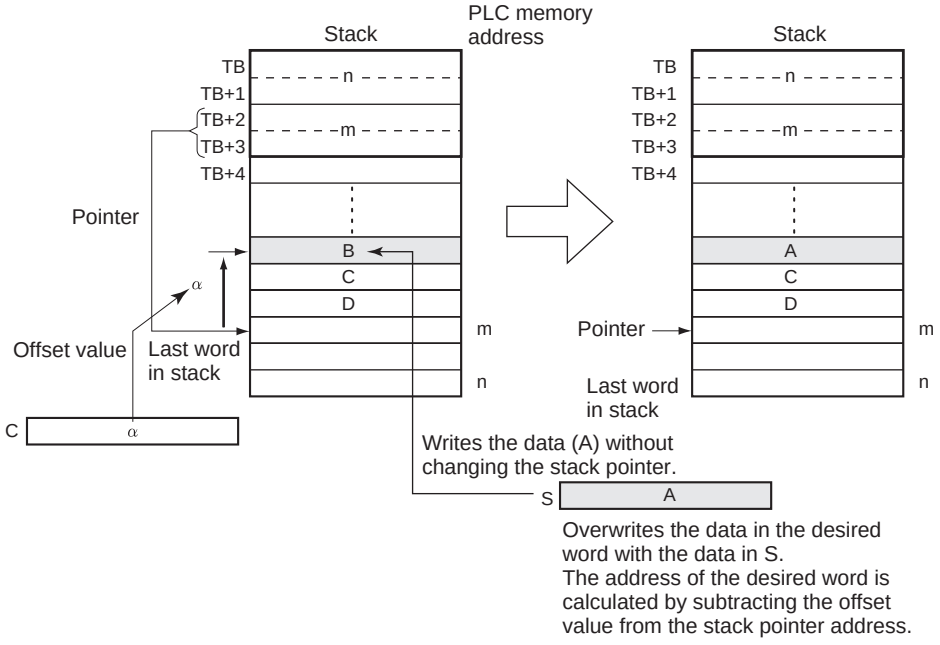


Operand Specifications

Area	TB	C	S
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0001 to #FFFB (Hexadecimal)	#0000 to #FFFF (Hexadecimal)
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

SWRIT(640) overwrites the data in the desired word with the data specified in S. The location of the desired word is calculated by subtracting the offset value in C from the stack pointer (TB+3 and TB+2). SWRIT(640) does not change the stack pointer.



SWRIT(640) can be used to change the data for an item currently on a conveyor. The position of the desired item is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified write location is not within the stack area. ON if the offset value specified in C is 0 or greater than the maximum data region size (FFFB hex). OFF in all other cases.

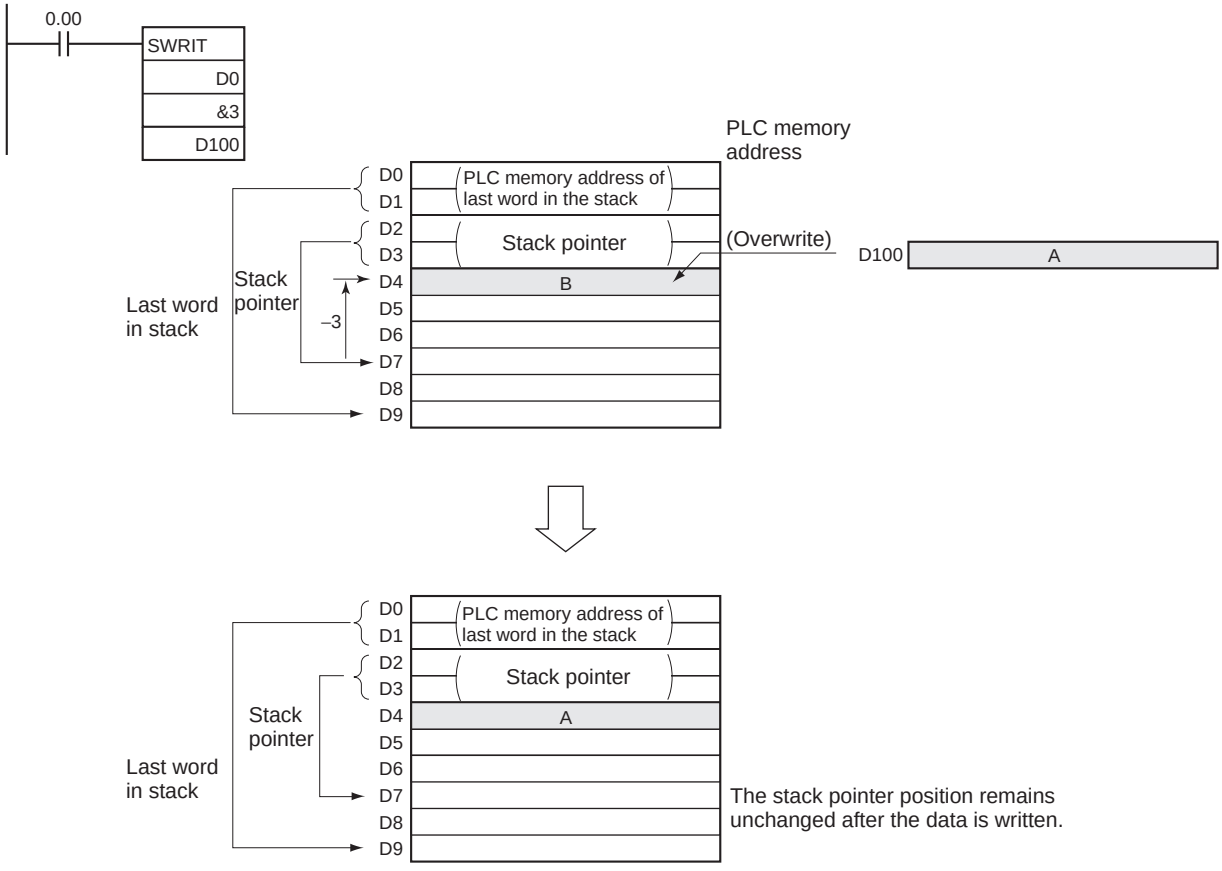
Precautions

The stack must be defined in advance with SSET(630).

The address in the stack pointer must be greater than the PLC memory address of the beginning of the data region (TB+4). An error will occur if the stack pointer is less than the PLC memory address of TB+4, i.e., if a stack underflow error occurs.

Examples

When CIO 0.00 is ON in the following example, SWRIT(640) writes the data in D100 to the specified word in the stack starting at D0. In this case, the stack pointer indicates D7 and the offset value is 3, so the data in D4 is overwritten.

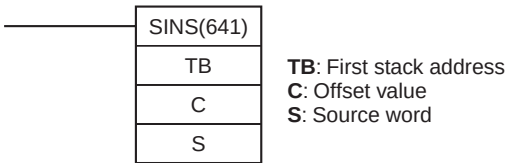


3-16-17 STACK DATA INSERT: SINS(641)

Purpose

Inserts the source data at the specified location in the stack and shifts the rest of the data in the stack downward. The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SINS(641)
	Executed Once for Upward Differentiation	@SINS(641)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

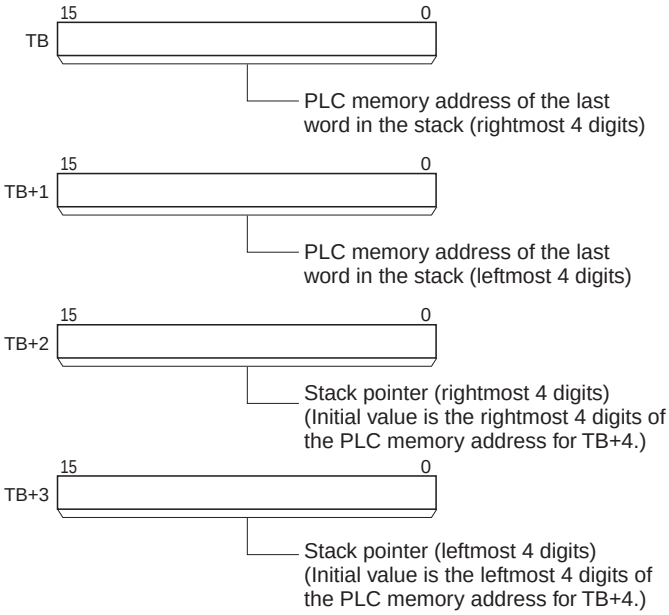
Applicable Program Areas

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

Operands

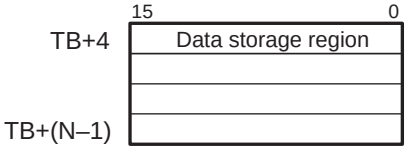
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

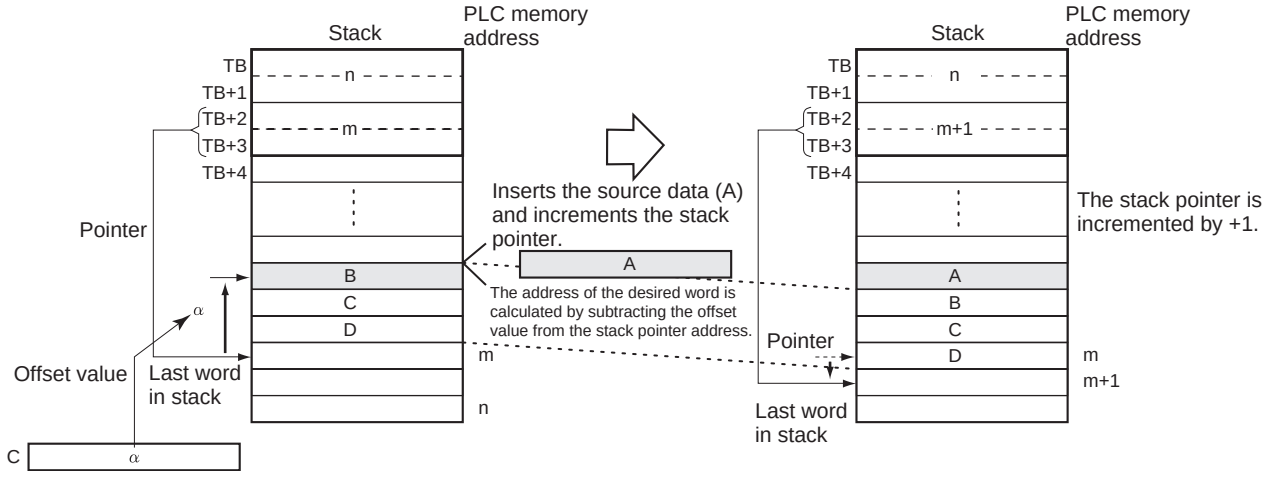


Operand Specifications

Area	TB	C	S
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0001 to #FFFB (Hexadecimal)	#0000 to #FFFF (Hexadecimal)
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

SINS(641) inserts the source data at the desired address and shifts the existing data down one word. At the same time, SINS(641) increments the stack pointer (TB+3 and TB+2) by 1. The location of the desired address is calculated by subtracting the offset value in C from the stack pointer.



SINS(641) can be used to insert the data for an item that is inserted in the midst of items already on a conveyor. The position of the insertion point is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the address indicated by the stack pointer (TB+3 and TB+2) is greater than the PLC memory address of last word in the data region of the stack. (This is a stack overflow error.) ON if the offset value specified is greater than the maximum data region size - 1 (FFFA hex). OFF in all other cases.

Precautions

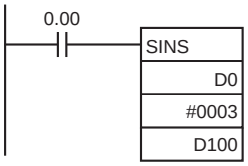
The stack must be defined in advance with SSET(630).

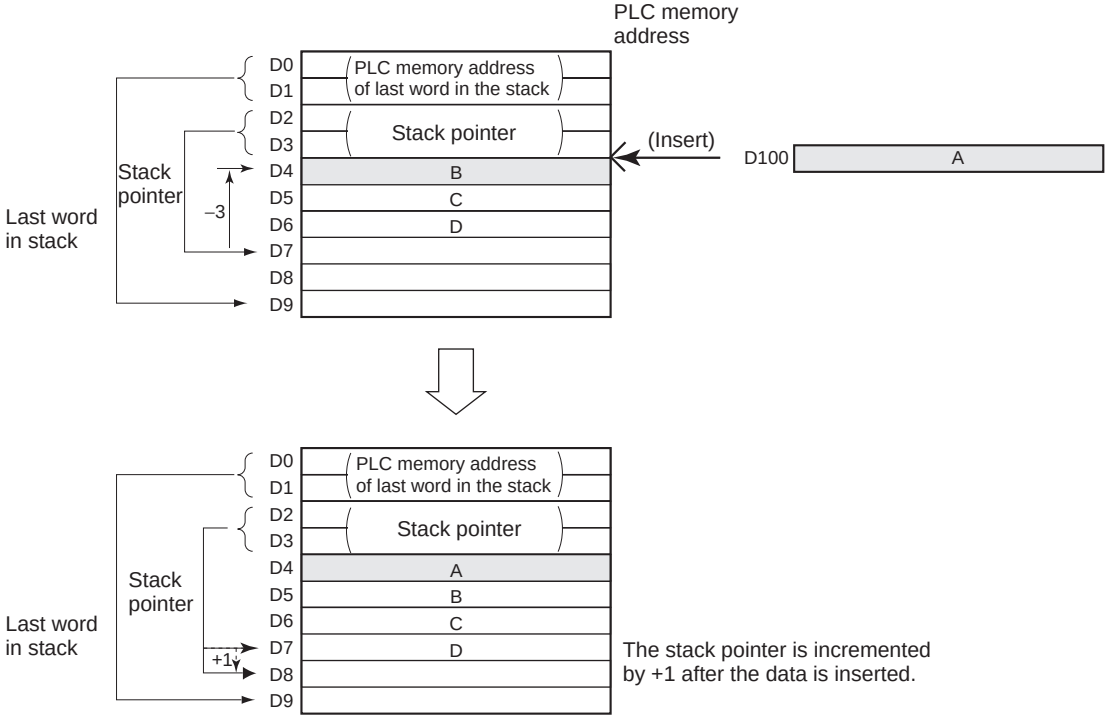
SINS(641) inserts one word of data into the stack, so there must be at least one available word at the end of the stack. If the stack is full, an error will occur and the source data will not be inserted.

If the address indicated by the stack pointer (TB+3 and TB+2) is already greater than the address of the last word in the stack (TB+1 and TB) when SINS(641) is executed, a stack overflow error will occur and the source data will not be inserted.

Examples

When CIO 0.00 is ON in the following example, SINS(641) inserts the source data in D100 at the specified address in the stack starting at D0. In this case, the stack pointer indicates D7 and the offset value is 3, so the source data is inserted in D4. The existing data is shifted down one word and the data in D7 is overwritten. At the same time the stack pointer will be incremented from D7 to D8.



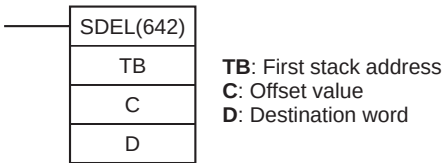


3-16-18 STACK DATA DELETE: SDEL(642)

Purpose

Deletes the data element at the specified location in the stack, outputs that data to the specified destination word, and shifts the remaining the data in the stack upward. The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SDEL(642)
	Executed Once for Upward Differentiation	@SDEL(642)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

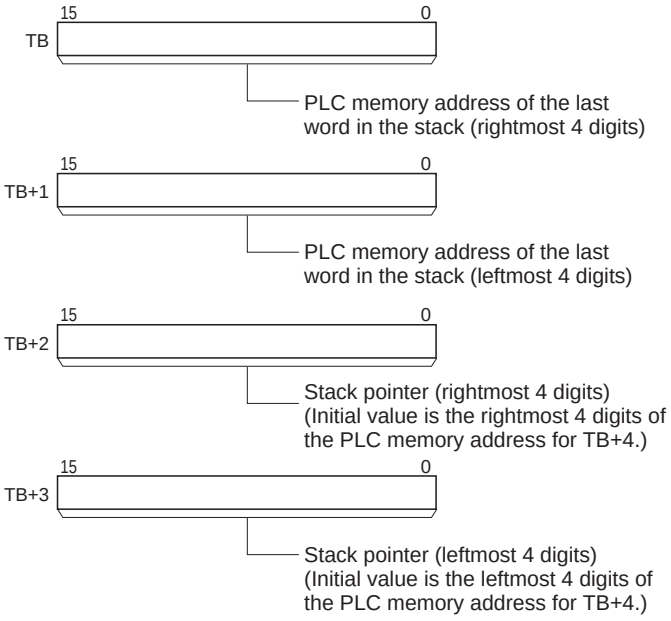
Applicable Program Areas

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

Operands

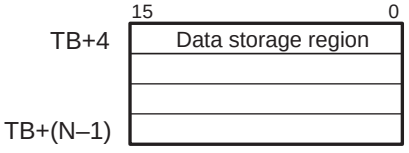
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

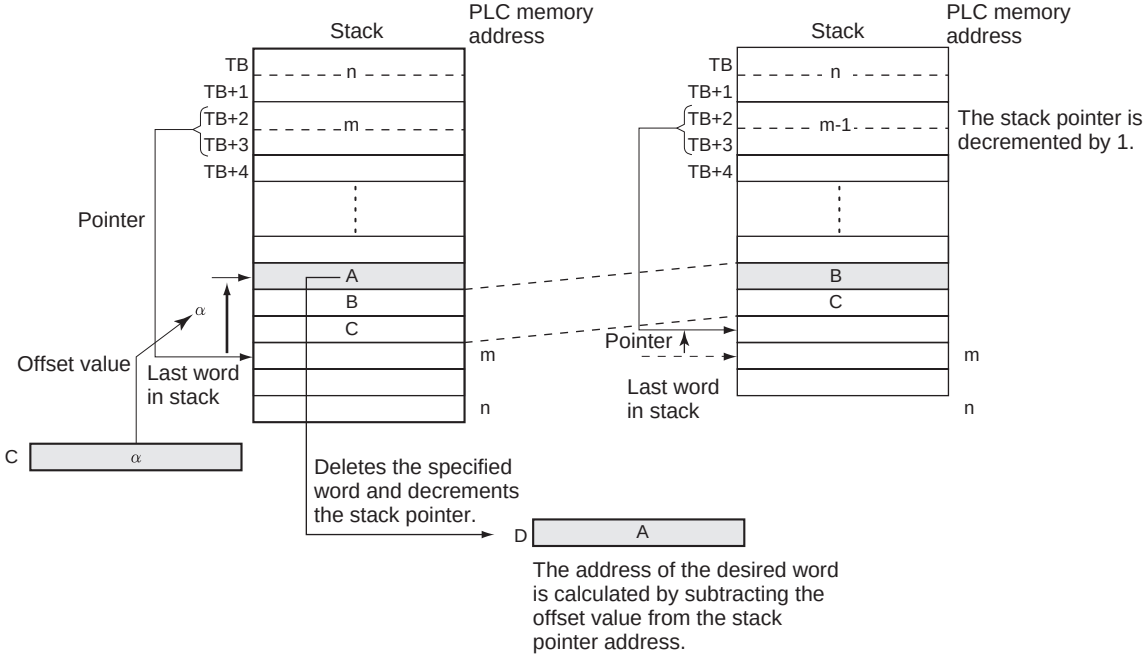


Operand Specifications

Area	TB	C	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A959	A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0001 to #FFFB (Hexadecimal)	---
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

SDEL(642) deletes the data at the specified location in the stack, outputs that data to the specified destination word, and shifts the remaining the data in the stack upward. At the same time, SDEL(642) decrements the stack pointer (TB+3 and TB+2) by 1. The location of the desired address is calculated by subtracting the offset value in C from the stack pointer.



SDEL(642) can be used to delete the data for an item that is rejected from the items on a conveyor. The position of the deletion point is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of the stack pointer (TB+3 and TB+2) is less than or equal to the PLC memory address of first word in the data region of the stack (TB+4). (This is a stack underflow error.) ON if the offset value specified in C is 0 or greater than the maximum data region size (FFFB hex). OFF in all other cases.
Equals Flag	=	ON if the output data in D is 0000. OFF in all other cases.

Precautions

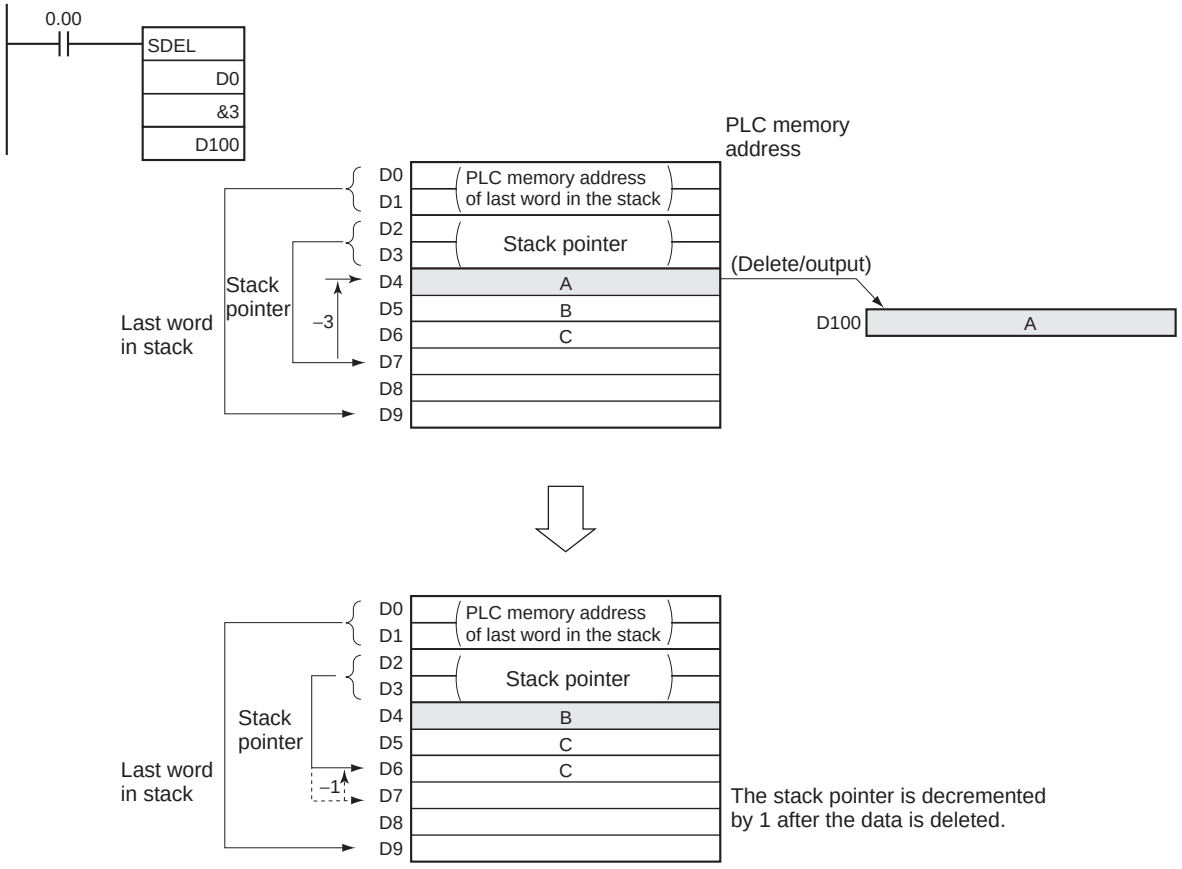
The stack must be defined in advance with SSET(630).

The address in the stack pointer must be greater than the PLC memory address of the beginning of the data region (TB+4). An error will occur if the stack pointer is less than the PLC memory address of TB+4, i.e., if a stack underflow error occurs.

Examples

When CIO 0.00 is ON in the following example, SDEL(642) deletes the word at the specified address in the stack starting at D0, outputs the deleted data to D100, shifts the remaining data upward, and decrements the stack pointer.

In this case, the stack pointer indicates D7 and the offset value is 3, so the data is deleted from D4. The remaining data is shifted up one word and the stack pointer is decremented from D7 to D6.



3-17 Data Control Instructions

This section describes instructions used to control specific operations.

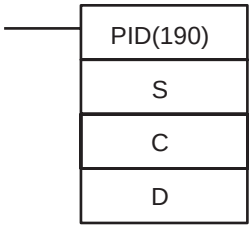
Instruction	Mnemonic	Function code	Page
PID CONTROL	PID	190	616
PID CONTROL WITH AUTOTUNING	PIDAT	191	628
LIMIT CONTROL	LMT	680	638
DEAD BAND CONTROL	BAND	681	640
DEAD ZONE CONTROL	ZONE	682	643
TIME-PROPORTIONAL OUTPUT	TPO	685	645
SCALING	SCL	194	653
SCALING 2	SCL2	486	657
SCALING 3	SCL3	487	661
AVERAGE	AVG	195	665

3-17-1 PID CONTROL: PID(190)

Purpose

Executes PID control according to the specified parameters.

Ladder Symbol



S: Input word

C: First parameter word

D: Output word

Variations

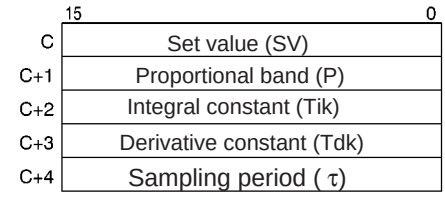
Variations	**Executed Each Cycle for ON Condition**	PID(190)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

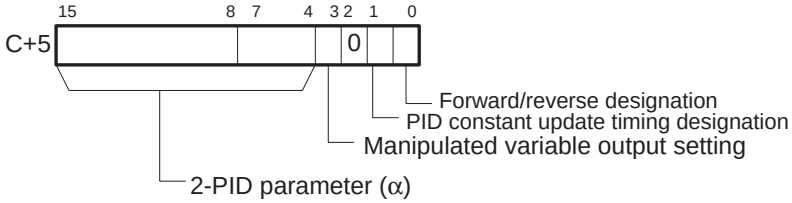
Applicable Program Areas

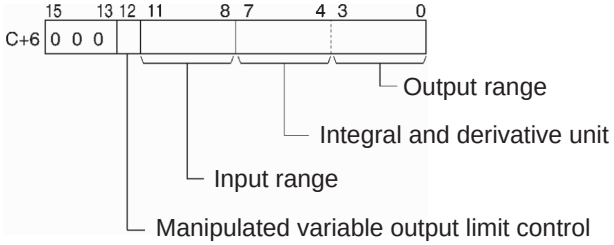
Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	Not allowed

Parameters

The following diagrams show the locations of the parameter data. For details on the parameters, refer to *PID Parameter Settings* in this section.







Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6105	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W473	W0 to W511
Holding Bit Area	H0 to H511	H0 to H473	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A921	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4057	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4057	C0000 to C4095
DM Area	D0 to D32767	D0 to D32729	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	DR0 to DR15	---	DR0 to DR15
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 IR15		

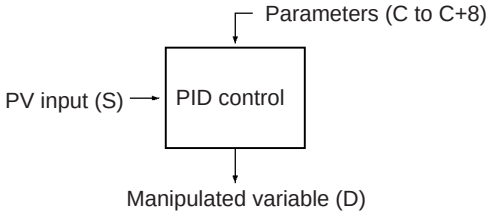
Description

When the execution condition is ON, PID(190) carries out target value filtered PID control with two degrees of freedom according to the parameters designated by C (set value, PID constant, etc.). It takes the specified input range of binary data from the contents of input word S and carries out the PID action according to the parameters that are set. The result is then stored as the manipulated variable in output word D.

The parameters are obtained when the execution condition turns from OFF to ON, and the Error Flag will turn ON if the settings are outside of the permissible range.

If the settings are within the permissible range, PID processing will be executed using the initial values. Bumpless operation is not performed at this time. It will be used for manipulated variables in subsequent PID processing execution. (Bumpless operation is processing that gradually and continuously changes the manipulated variable in order to avoid the adverse effects of sudden changes.)

When the execution condition turns ON, the PV for the specified sampling period is entered and processing is performed.



The number of valid input data bits within the 16 bits of the PV input (S) is designated by the input range setting in C+6, bits 08 to 11. For example, if 12 bits (4 hex) is designated for the input range, the range from 0000 hex to 0FFF hex will be enabled as the PV. (Values greater than 0FFF hex will be regarded as 0FFF hex.)

The set value range also depends on the input range.

Measured values (PV) and set values (SV) are in binary without sign, from 0000 hex to the maximum value of the input range.

The number of valid output data bits within the 16 bits of the manipulated variable output is designated by the output range setting in C+6, bits 00 to 03. For example, if 12 bits (4 hex) is designated for the output range, the range from 0000 hex to 0FFF hex will be output as the manipulated variable.

For proportional operation only, the manipulated variable output when the PV equals the SV can be designated as follows:

0: Output 0%
1: Output 50%.

The direction of proportional operation can be designated as either forward or reverse.

The upper and lower limits of the manipulated variable output can be designated.

The sampling period can be designated in units of 10 ms (0.01 to 99.99 s), but the actual PID action is determined by a combination of the sampling period and the time of PID(190) instruction execution (with each cycle).

The timing of enabling changes made to PID constants can be set to either 1) the beginning of PID instruction execution or 2) the beginning of PID instruction execution and each sampling period. Only the proportional band (P), integral constant (Tik), and derivative constant (Tdk) can be changed each sampling cycle (i.e., during PID instruction execution). The timing is set in bit 1 of C+5.

Of the PID parameters (C to C+38), only the set value (SV) can be changed when the execution condition is ON. When changing other values, be sure to change the execution condition from OFF to ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the C data is out of range. ON if the actual sampling period is more than twice the designated sampling period. OFF in all other cases.
Greater Than Flag	>	ON if the manipulated variable after the PID action exceeds the upper limit. OFF in all other cases.
Less Than Flag	<	ON if the manipulated variable after the PID action is below the lower limit. OFF in all other cases.
Carry Flag	CY	ON while PID control is being executed. OFF in all other cases.

Precautions

PID(190) is executed as if the execution condition was a STOP-RUN signal. PID calculations are executed when the execution condition remains ON for the next cycle after C+9 to C+38 are initialized. Therefore, when using the Always ON Flag (ON) as an execution condition for PID(190), provide a separate process where C+9 to C+38 are initialized when operation is started.

If the C data is out of range, an error will occur and the Error Flag will turn ON.

If the actual sampling period is more than twice the designated sampling period, an error will occur and the Error Flag will turn ON. PID control will still be executed, however.

The Carry Flag turns ON while PID control is being executed.

The Greater Than Flag turns ON if the manipulated variable after the PID action exceeds the upper limit. At this time, the results are output at the upper limit.

The Less Than Flag turns ON if the manipulated variable after the PID action is below the lower limit. At this time, the results are output at the lower limit.

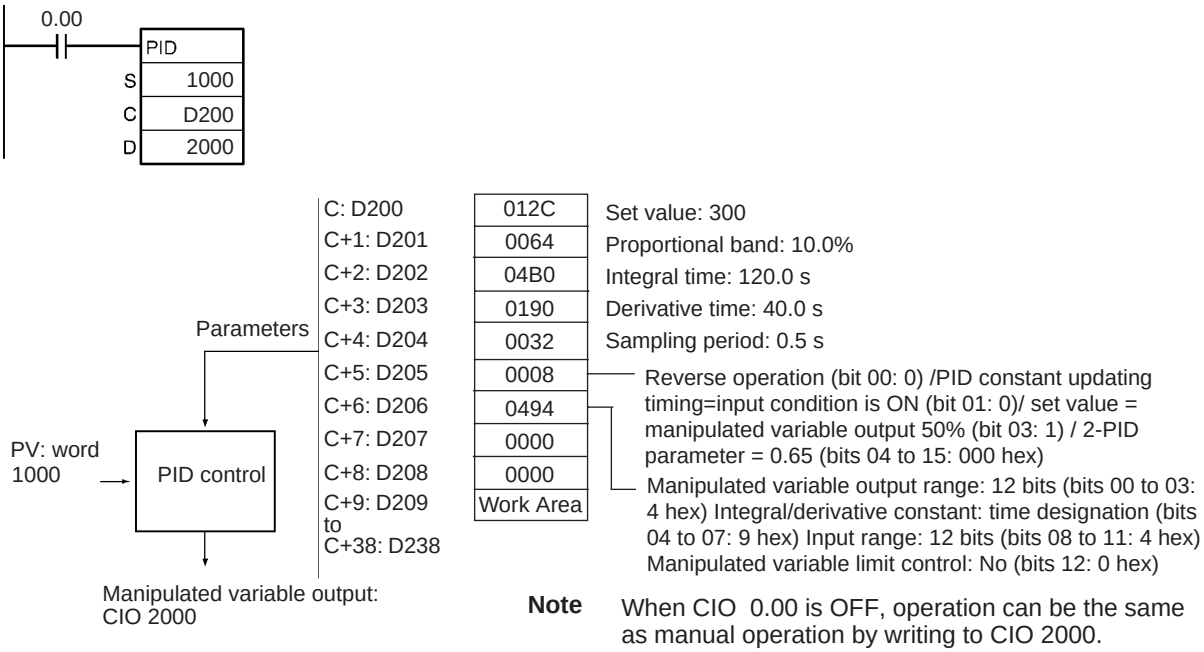
Within the PID parameters (C to C+38), the only value that can be changed while the input condition is ON is the set value for C. If any other value is changed, be sure to turn the input condition from OFF to ON to enable the new value.

Example

At the rising edge of CIO 0.00 (OFF to ON), the work area in D209 to D238 is initialized according to the parameters (shown below) set in D200 to D208. After the work area has been initialized, PID control is executed and the manipulated variable is output to CIO 2000.

When CIO 0.00 is turned ON, PID control is executed at the sampling period intervals according to the parameters set in D200 to D208. The manipulated variable is output to CIO 2000.

The PID constants used in PID calculations will not be changed if the proportional band (P), integral constant (Tik), or derivative constant is changed after CIO 0.00 turns ON.



Note When CIO 0.00 is OFF, operation can be the same as manual operation by writing to CIO 2000.

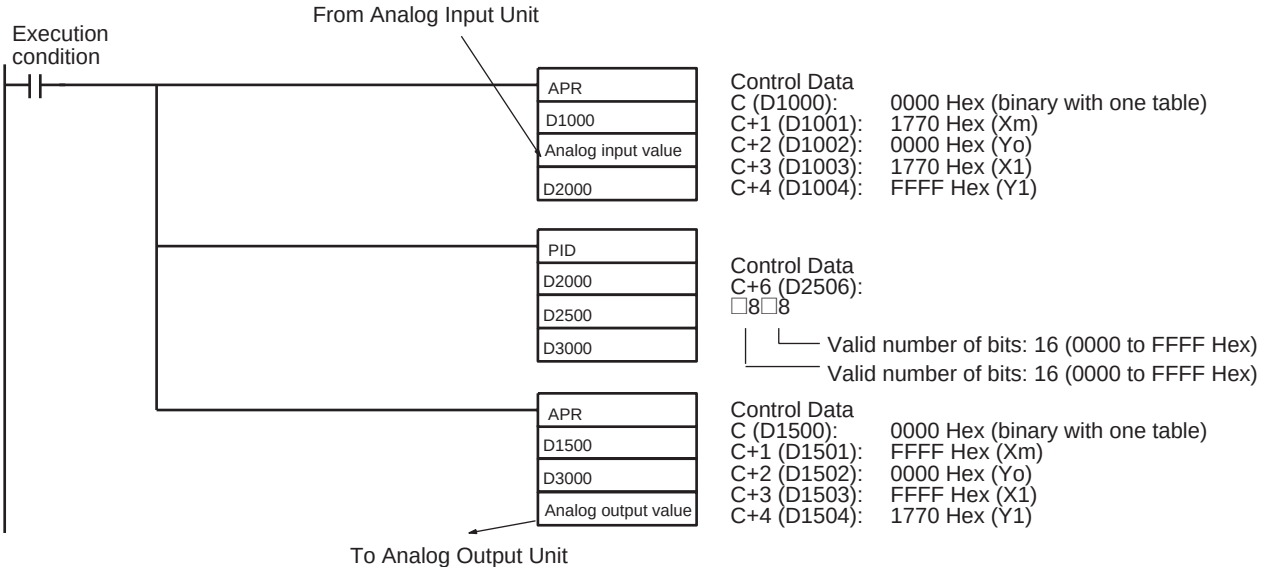
Input Values and Manipulated Variable Ranges

The number of valid input data bits for the measured value is designated by the input range setting in C+6, bits 08 to 11, and the number of valid output data bits for the manipulated variable output is designated by the output range setting in C+6, bits 0 to 3. These ranges are shown in the following table.

C+6, bits 08 to 11 or C+6, bits 00 to 03	Number of valid bits	Range
0	8	0000 to 00FF hex
1	9	0000 to 01FF hex
2	10	0000 to 03FF hex
3	11	0000 to 07FF hex
4	12	0000 to 0FFF hex
5	13	0000 to 1FFF hex
6	14	0000 to 3FFF hex
7	15	0000 to 7FFF hex
8	16	0000 to FFFF hex

If the range of data handled by an Analog Input Unit or Analog Output Unit cannot be set accurately by setting the number of valid bits, APR(069) (ARITHMETIC PROCESS) can be used to convert to the proper ranges before and after PID(190).

The following program section shows an example for a DRT1-AD04 Analog Input Unit and DRT1-DA02 Analog Output Unit operating as DeviceNet slaves. The data ranges for these two Units is 0000 to 1770 hex, which cannot be specified merely by setting the valid number of digits. APR(069) is thus used to convert the 0000 to 1770 hex range of the Analog Input Unit to 0000 to FFFF hex for input to PID(190) and then the manipulated variable output from PID(190) is converted back to the range 0000 to 1770 hex, again using APR(069), for output from the Analog Output Unit.

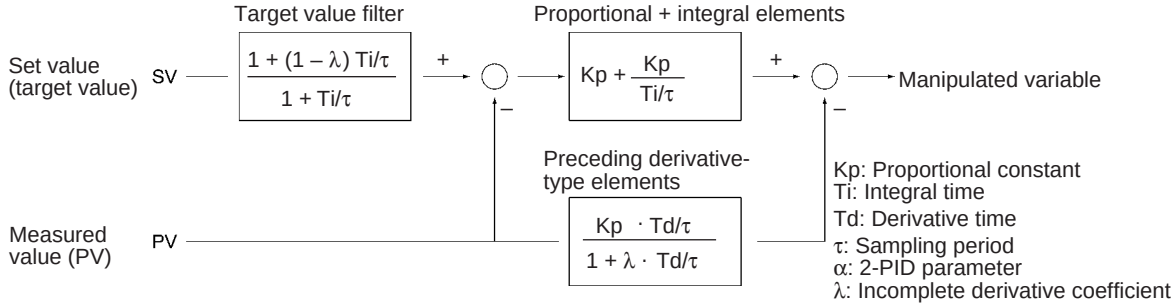


Performance Specifications

Item			Specifications
PID control method		---	Target value filter-type two-degrees-of-freedom PID method (forward/reverse)
Number of PID control loops		---	Unlimited (1 loop per instruction)
Sampling period		τ	0.01 to 99.99 s
PID constant	Proportional band	P	0.1 to 999.9%
	Integral constant	Tik	1 to 8191, 9999 (No integral action for sampling period multiple, 9999.)
	Derivative constant	Tdk	0 to 8191 (No derivative action for sampling period multiple, 0.)
Set value		SV	0 to 65535 (Valid up to maximum value of input range.)
Measured value		PV	0 to 65535 (Valid up to maximum value of input range.)
Manipulated variable		MV	0 to 65535 (Valid up to maximum value of output range.)

Calculation Method

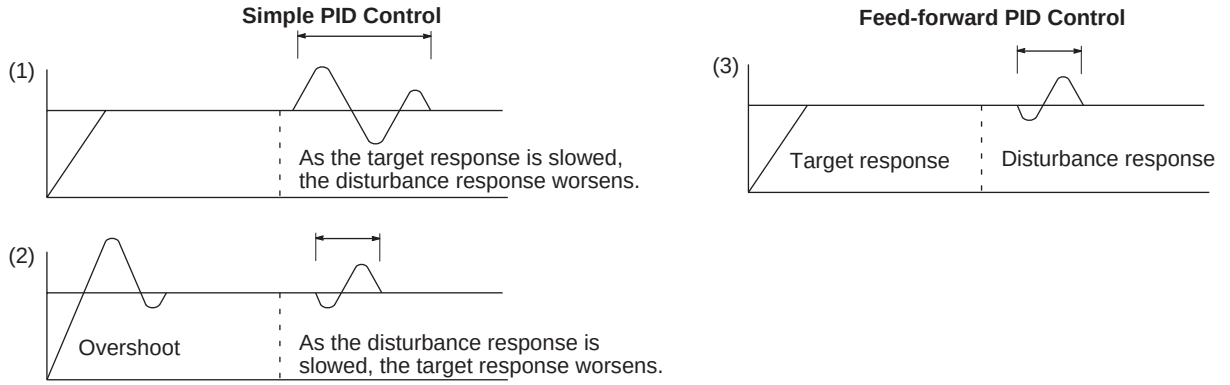
Calculations in PID control are performed by the target value filtered control with two degrees of freedom.



Block Diagram for Target Value PID with Two Degrees of Freedom

When overshooting is prevented with simple PID control, stabilization of disturbances is slowed (1). If stabilization of disturbances is speeded up, on the other hand, overshooting occurs and response toward the target value is slowed (2).

When target-value PID control with two degrees of freedom is used, on the other hand, there is no overshooting, and response toward the target value and stabilization of disturbances can both be speeded up (3).



PID Parameter Settings

Control data	Item	Contents	Setting range	Change with ON input condition
C	Set value (SV)	The target value of the process being controlled.	Binary data (of the same number of bits as specified for the input range)	Allowed
C+1	Proportional band	The parameter for P action expressing the proportional control range/total control range.	0001 to 270F hex (1 to 9999); (0.1% to 999.9%, in units of 0.1%)	Can be changed with input condition ON if bit 1 of C+5 is 1.
C+2	Tik Integral Constant	A constant expressing the strength of the integral action. As this value increases, the integral strength decreases.	0001 to 1FFF hex (1 to 8191); (9999 = Integral operation not executed) (See note 1.)	
C+3	Tdk Derivative Constant	A constant expressing the strength of the derivative action. As this value increases, the derivative strength decreases.	0001 to 1FFF hex (1 to 8191); (0000 = Derivative operation not executed) (See note 1.)	
C+4	Sampling period (τ)	Sets the period for executing the PID action.	0001 to 270F hex (1 to 9999); (0.01 to 99.99 s, in units of 10 ms)	Not allowed
Bits 04 to 15 of C+5	2-PID parameter (α)	The input filter coefficient. Normally use 0.65 (i.e., a setting of 000). The filter efficiency decreases as the coefficient approaches 0.	000 hex: α = 0.65 Setting from 100 to 163 hex means that the value of the rightmost two digits is set from α= 0.00 to α= 0.99. (See note 2.)	
Bit 03 of C+5	Manipulated variable output designation	Designates the manipulated variable output for when the PV equals the SV.	0: Output 0% 1: Output 50%	
Bit 01 of C+5	PID constant change enable setting	The timing of enabling changes made to the proportional band (P), integral constant (Tik), and derivative constant (Tdk) for use in PID calculations.	0: At start of PID instruction execution 1: At start of PID instruction execution and each sampling period	Allowed

Control data	Item	Contents	Setting range	Change with ON input condition
Bit 00 of C+5	PID forward/reverse designation	Determines the direction of the proportional action.	0: Reverse action 1: Forward action	Not allowed
Bit 12 of C+6	Manipulated variable output limit control	Determines whether or not limit control will apply to the manipulated variable output.	0: Disabled (no limit control) 1: Enabled (limit control)	
Bits 08 to 11 of C+6	Input range	The number of input data bits.	0: 8 bits 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits7: 15 bits 3: 11 bits8: 16 bits 4: 12 bits	
Bits 04 to 07 of C+6	Integral and derivative unit	Determines the unit for expressing the integral and derivative constants.	1: Sampling period multiple 9: Time (unit: 100 ms)	
Bits 00 to 03 of C+6	Output range	The number of output data bits. (The number of output bits is automatically the same as the number of input bits.)	0: 8 bits 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits7: 15 bits 3: 11 bits8: 16 bits 4: 12 bits	
C +7	Manipulated variable output lower limit	The lower limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	
C +8	Manipulated variable output upper limit	The upper limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	

Note

(1) When the unit is designated as 1, the range is from 1 to 8,191 times the period. When the unit is designated as 9, the range is from 0.1 to 819.1 s. When 9 is designated, set the integral and derivative times to within a range of 1 to 8,191 times the sampling period.

(2) Setting the 2-PID parameter (α) to 000 yields 0.65, the normal value.

(3) When the manipulated variable output limit control is enabled (i.e., set to "1"), set the values as follows:

$0000 \leq$ MV output lower limit $\leq$ MV output upper limit $\leq$ Max. value of output range

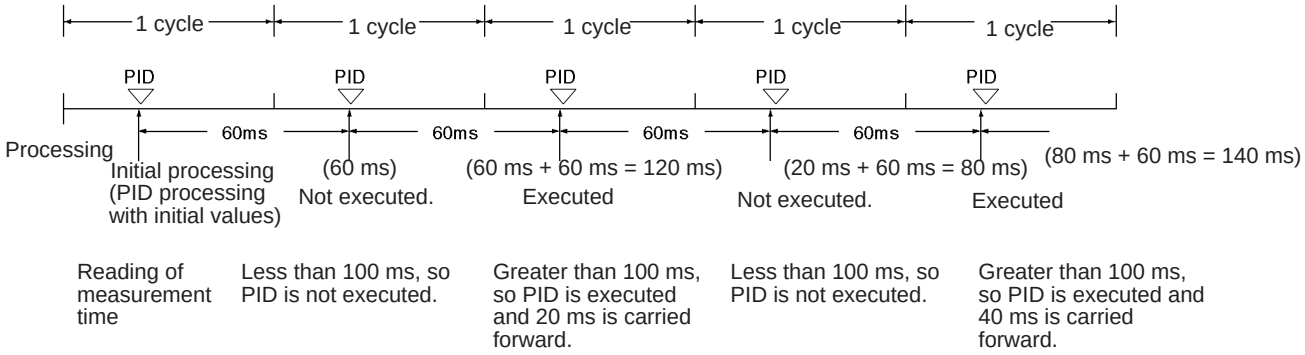
Sampling Period and Cycle Time

The sampling period can be designated in units of 10 ms (0.01 to 99.99 s), but the actual PID action is determined by a combination of the sampling period and the time of PID instruction execution (with each cycle). The relationship between the sampling period and the cycle time is as follows:

- If the sampling period is less than the cycle time, PID control is executed with each cycle and not with each sampling period.
- If the sampling period is greater than or equal to the cycle time, PID control is not executed with each cycle, but PID(190) is executed when the cumulative value of the cycle time (the time between PID instructions) is greater than or equal to the sampling period. The surplus portion of the cumulative value (i.e., the cycle time's cumulative value minus the sampling period) is carried forward to the next cumulative value.

  For example, suppose that the sampling period is 100 ms and that the cycle time is consistently 60 ms. For the first cycle after the initial execution, PID(190) will not be executed because 60 ms is less than 100 ms. For the second cycle, 60 ms + 60 ms is greater than 100 ms, so PID(190) will be executed. The surplus of 20 ms (i.e., 120 ms – 100 ms = 20 ms) will be carried forward.

For the third cycle, the surplus 20 ms is added to 60 ms. Because the sum of 80 ms is less than 100 ms, PID(190) will not be executed. For the fourth cycle, the 80 ms is added to 60 ms. Because the sum of 140 ms is greater than 100 ms, PID(190) will be executed and the surplus of 40 ms (i.e., 120 ms – 100 ms = 20 ms) will be carried forward. This procedure is repeated for subsequent cycles.



Control Actions

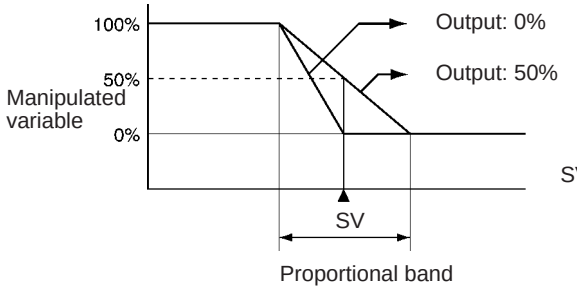
Proportional Action (P)

Proportional action is an operation in which a proportional band is established with respect to the set value (SV), and within that band the manipulated variable (MV) is made proportional to the deviation. An example for reverse operation is shown in the following illustration.

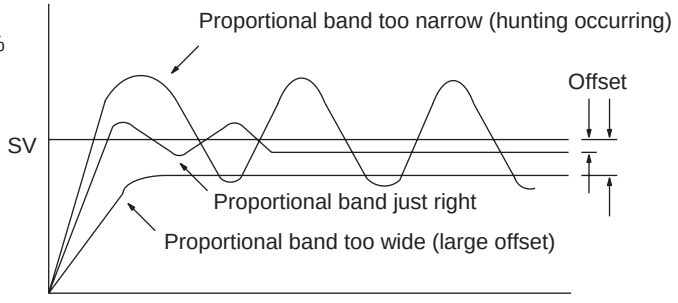
If the proportional action is used and the present value (PV) becomes smaller than the proportional band, the manipulated variable (MV) is 100% (i.e., the maximum value). Within the proportional band, the MV is made proportional to the deviation (the difference between from SV and PV) and gradually decreased until the SV and PV match (i.e., until the deviation is 0), at which time the MV will be at the minimum value of 0% (or 50%, depending on the setting of the manipulated variable output designation parameter). The MV will also be 0% when the PV is larger than the SV.

The proportional band is expressed as a percentage of the total input range. The smaller the proportional band, the larger the proportional constant and the stronger the corrective action will be. With proportional action an offset (residual deviation) generally occurs, but the offset can be reduced by making the proportional band smaller. If it is made too small, however, hunting will occur.

Proportional Action (Reverse Action)

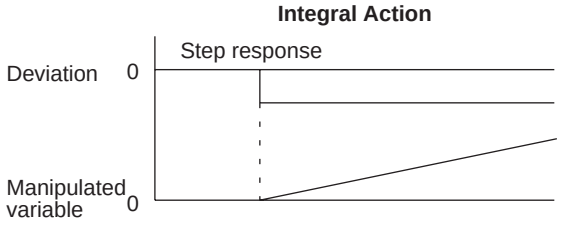


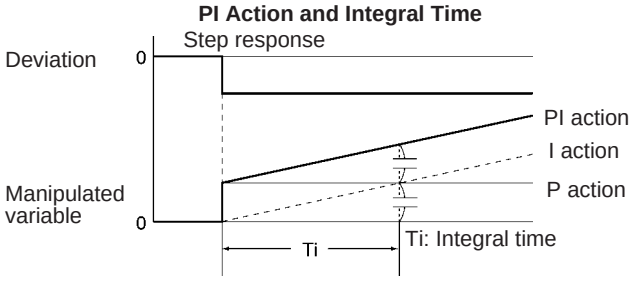
Adjusting the Proportional Band



Integral Action (I)

Combining integral action with proportional action reduces the offset according to the time that has passed, so that the PV will match the SV. The strength of the integral action is indicated by the integral time, which is the time required for the manipulated variable of the integral action to reach the same level as the manipulated variable of the proportional action with respect to the step deviation, as shown in the following illustration. The shorter the integral time, the stronger the correction by the integral action will be. If the integral time is too short, the correction will be too strong and will cause hunting to occur.



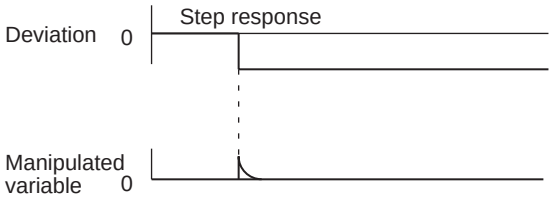


Derivative Action (D)

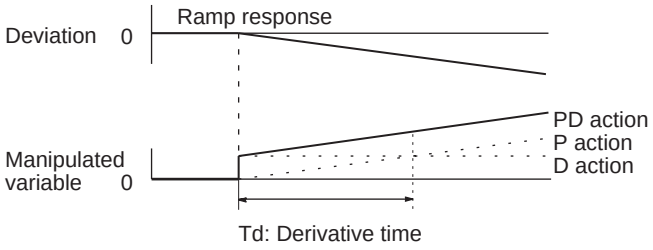
Proportional action and integral action both make corrections with respect to the control results, so there is inevitably a response delay. Derivative action compensates for that drawback. In response to a sudden disturbance it delivers a large manipulated variable and rapidly restores the original status. A correction is executed with the manipulated variable made proportional to the incline (derivative coefficient) caused by the deviation.

The strength of the derivative action is indicated by the derivative time, which is the time required for the manipulated variable of the derivative action to reach the same level as the manipulated variable of the proportional action with respect to the step deviation, as shown in the following illustration. The longer the derivative time, the stronger the correction by the derivative action will be.

Derivative Action



PD Action and Derivative Time

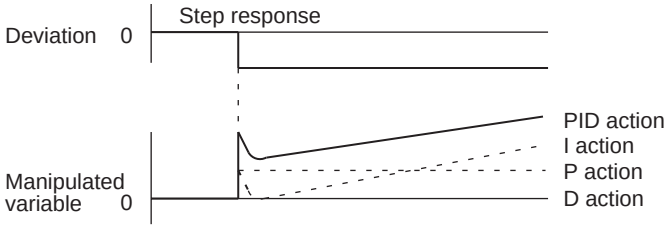


Td: Derivative time

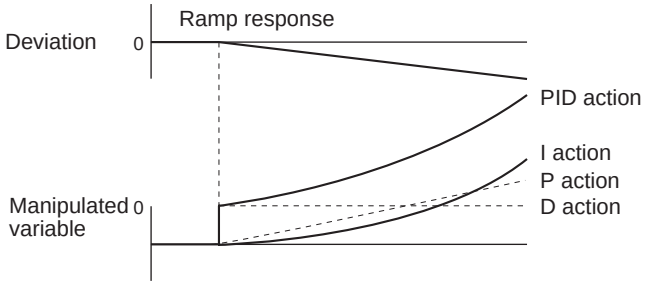
PID Action

PID action combines proportional action (P), integral action (I), and derivative action (D). It produces superior control results even for control objects with dead time. It employs proportional action to provide smooth control without hunting, integral action to automatically correct any offset, and derivative action to speed up the response to disturbances.

Step Response of PID Control Action Output



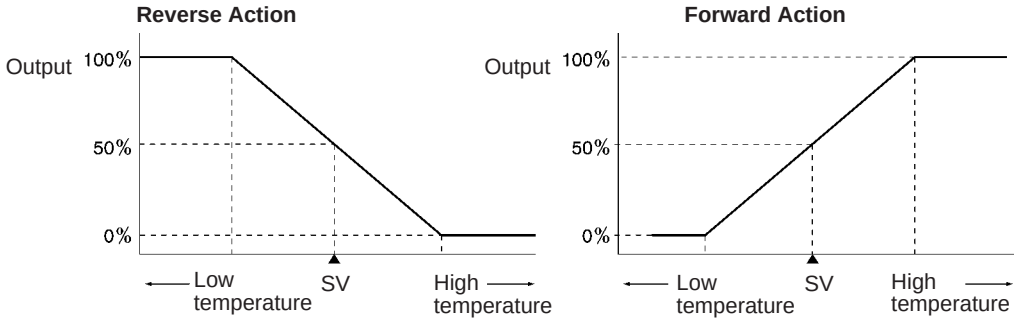
Ramp Response of PID Control Action Output



Direction of Action

When using PID control, select either of the following two control directions. In either direction, the MV increases as the difference between the SV and the PV increases.

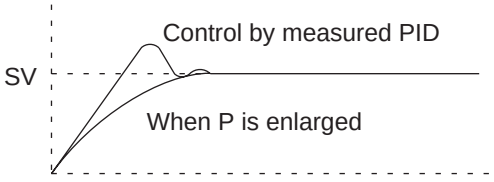
- Forward action: MV is increased when the PV is larger than the SV.
- Reverse action: MV is increased when the PV is smaller than the SV.



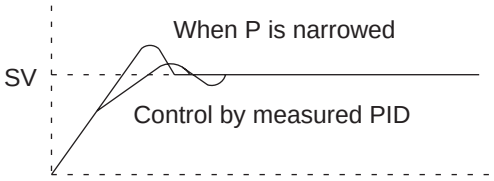
Adjusting PID Parameters

The general relationship between PID parameters and control status is shown below.

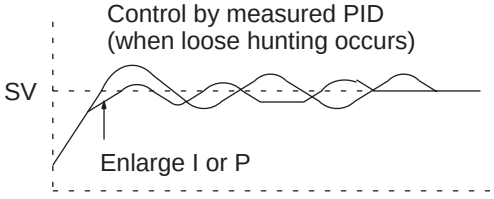
- When it is not a problem if a certain amount of time is required for stabilization (settlement time), but it is important not to cause overshooting, then enlarge the proportional band.



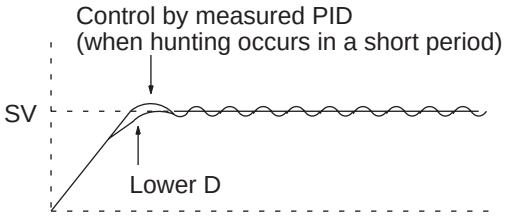
- When overshooting is not a problem but it is desirable to quickly stabilize control, then narrow the proportional band. If the proportional band is narrowed too much, however, then hunting may occur.



- When there is broad hunting, or when operation is tied up by overshooting and undershooting, it is probably because integral action is too strong. The hunting will be reduced if the integral time is increased or the proportional band is enlarged.



- If the period is short and hunting occurs, it may be that the control system response is quick and the derivative action is too strong. In that case, set the derivative action lower.

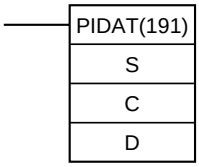


3-17-2 PID CONTROL WITH AUTOTUNING: PIDAT(191)

Purpose

Executes PID control according to the specified parameters. The PID constants can be autotuned.

Ladder Symbol



S: Input word
C: First parameter word
D: Output word

Variations

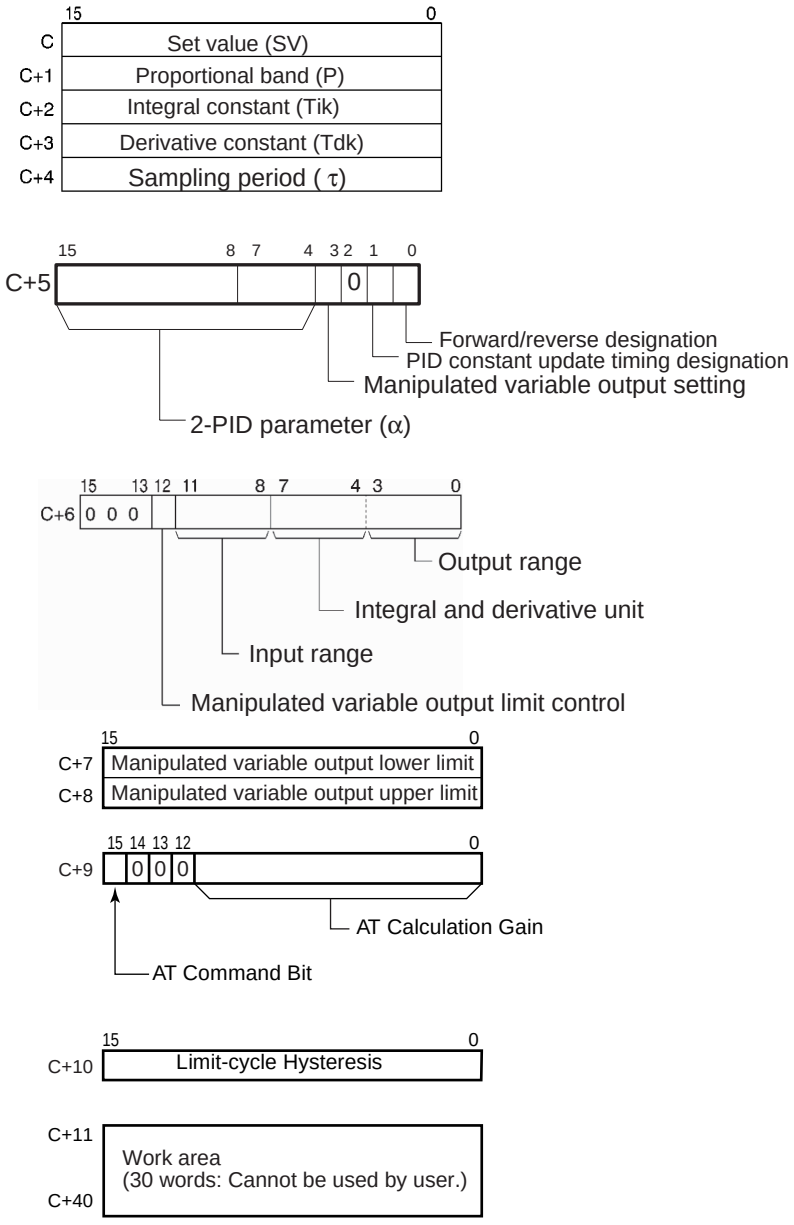
Variations	**Executed Each Cycle for ON Condition**	PIDAT(191)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Parameters

The following diagrams show the locations of the parameter data. For details on the parameters, refer to *PID Parameter Settings* in this section.



Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6105	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W473	W0 to W511
Holding Bit Area	H0 to H511	H0 to H473	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A921	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4057	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4057	C0000 to C4095
DM Area	D0 to D32767	D0 to D32729	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	DR0 to DR15	---	DR0 to DR15
Data Registers	---		

Area	S	C	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 IR15		

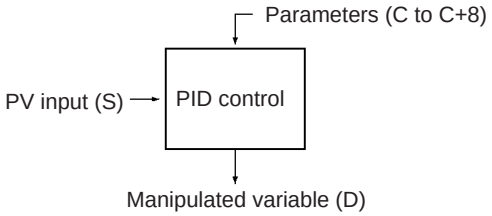
Description

When the execution condition is ON, PIDAT(191) carries out target value filtered PID control with two degrees of freedom according to the parameters designated by C (set value, PID constant, etc.). It takes the specified input range of binary data from the contents of input word S and carries out the PID action according to the parameters that are set. The result is then stored as the manipulated variable in output word D.

The parameter settings are read when the execution condition turns from OFF to ON, and the Error Flag will turn ON if the settings are outside of the permissible range.

If the settings are within the permissible range, PID processing will be executed using the initial values. Bumpless operation is not performed at this time. It will be used for manipulated variables in subsequent PID processing execution. (Bumpless operation is processing that gradually and continuously changes the manipulated variable in order to avoid the adverse effects of sudden changes.)

When the execution condition turns ON, the PV for the specified sampling period is entered and processing is performed.



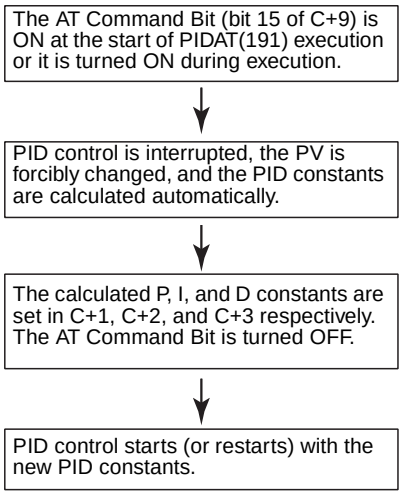
Autotuning

The status of the AT Command Bit (bit 15 of C+9) is checked every cycle. If this control bit is turned ON in a given cycle, PIDAT(191) will begin autotuning the PID constants. (The changes in the SV will not be reflected while autotuning is being performed.)

The limit-cycle method is used for autotuning. PIDAT(191) forcibly changes the manipulated variable (max. manipulated variable ↔ min. manipulated variable) and monitors the characteristics of the controlled system. The PID constants are calculated based on the characteristics that were observed, and the new P, I, and D constants are stored automatically in C+1, C+2, and C+3. At this point, the AT Command Bit (bit 15 of C+9) is turned OFF and PID control resumes with the new PID constants in C+1, C+2, and C+3.

- If the AT Command Bit is ON when PIDAT(191) execution begins, autotuning will be performed first and then PID control will start with the calculated PID constants.
- If the AT Command Bit is turned ON during PIDAT(191) execution, PIDAT(191) interrupts the PID control being performed with the user-set PID constants, performs autotuning, and then resumes PID control with the calculated PID constants.

The following flowchart shows the autotuning procedure:



Note

(1) If autotuning is interrupted by turning OFF the AT Command Bit during autotuning, PID control will start with the PID constants that were being used before autotuning began.

(2) Also, if an AT execution error occurs, PID control will start with the PID constants that were being used before autotuning began.

In both cases described in notes 1 and 2, the PID constants will be enabled if they were already calculated when autotuning was interrupted.

PID Control

The number of valid input data bits within the 16 bits of the PV input (S) is designated by the input range setting in C+6, bits 08 to 11. For example, if 12 bits (4 hex) is designated for the input range, the range from 0000 hex to 0FFF hex will be enabled as the PV. (Values greater than 0FFF hex will be regarded as 0FFF hex.)

The set value range also depends on the input range.

Measured values (PV) and set values (SV) are in binary without sign, from 0000 hex to the maximum value of the input range.

The number of valid output data bits within the 16 bits of the manipulated variable output is designated by the output range setting in C+6, bits 00 to 03. For example, if 12 bits (4 hex) is designated for the output range, the range from 0000 hex to 0FFF hex will be output as the manipulated variable.

For proportional operation only, the manipulated variable output when the PV equals the SV can be designated as follows:

0: Output 0%
1: Output 50%.

The direction of proportional operation can be designated as either forward or reverse.

The upper and lower limits of the manipulated variable output can be designated.

The sampling period can be designated in units of 10 ms (0.01 to 99.99 s), but the actual PID action is determined by a combination of the sampling period and the time of PIDAT(191) instruction execution (with each cycle).

The timing of enabling changes made to PID constants can be set to either 1) the beginning of PIDAT(191) instruction execution or 2) the beginning of PID instruction execution and each sampling period. Only the proportional band (P), integral constant (Tik), and derivative constant (Tdk) can be changed each sampling cycle (i.e., during PID instruction execution). The timing is set in bit 1 of C+5.

When changing the PID constants manually, set the PID constant change enable setting (bit 1 of C+5) to 1 so that the values in C+1, C+2, and C+3 are refreshed each sampling period in the PID calculation. This setting also allows the PID constants to be adjusted manually after autotuning.

Of the PID parameters (C to C+38), only the following parameters can be changed when the execution condition is ON. When any other values have been changed, be sure to change the execution condition from OFF to ON to enable the new settings.

- Set value (SV) in C
  (Can be changed during PID control only. An SV change during autotuning will not be reflected.)
- PID constant change enable setting (bit 1 of C+5)
- P, I, and D constants in C+1, C+2, and C+3
  (Changes to these constants will be reflected each sampling period only if the PID constant change enable setting (bit 1 of C+5) is set to 1.)
- AT Command Bit (bit 15 of C+9)
- AT Calculation Gain (bits 0 to 14 of C+9) and Limit-cycle Hysteresis (C+10) (These values are read when autotuning starts.)

Note The PIDAT(191) instruction is the same as the PID(190) instruction with the added autotuning (AT) function, so the PID control operations are identical. Refer to *3-17-1 PID CONTROL: PID(190)* for details on PID control operations and examples.

Flags

Name	Label	Operation
Error Flag	ER	ON if the C data is out of range. ON if the actual sampling period is more than twice the designated sampling period. ON if an error occurred during autotuning. OFF in all other cases.
Greater Than Flag	>	ON if the manipulated variable after the PID action exceeds the upper limit. OFF in all other cases.
Less Than Flag	<	ON if the manipulated variable after the PID action is below the lower limit. OFF in all other cases.
Carry Flag	CY	ON while PID control is being executed. OFF in all other cases.

Precautions

PIDAT(191) is executed as if the execution condition was a STOP-RUN signal. PID calculations are executed when the execution condition remains ON for the next cycle after C+11 to C+40 are initialized. Therefore, when using the Always ON Flag (ON) as an execution condition for PIDAT(191), provide a separate process where C+11 to C+40 are initialized when operation is started.

If the C data is out of range, an error will occur and the Error Flag will turn ON.

If an error occurred during autotuning, the Error Flag will turn ON.

If the actual sampling period is more than twice the designated sampling period, an error will occur and the Error Flag will turn ON. PID control will still be executed, however.

The Carry Flag turns ON while PID control is being executed.

The Greater Than Flag turns ON if the manipulated variable after the PID action exceeds the upper limit. At this time, the results are output at the upper limit.

The Less Than Flag turns ON if the manipulated variable after the PID action is below the lower limit. At this time, the results are output at the lower limit.

PID Parameter Settings

Control data	Item	Contents	Setting range	Change with ON input condition
C	Set value (SV)	The target value of the process being controlled.	Binary data (of the same number of bits as specified for the input range)	Allowed
C+1	Proportional band	The parameter for P action expressing the proportional control range/total control range.	0001 to 270F hex (1 to 9999); (0.1% to 999.9%, in units of 0.1%)	Can be changed with input condition ON if bit 1 of C+5 is 1.
C+2	Tik Integral Constant	A constant expressing the strength of the integral action. As this value increases, the integral strength decreases.	0001 to 1FFF hex (1 to 8191); (9999 = Integral operation not executed) (See note 1.)	
C+3	Tdk Derivative Constant	A constant expressing the strength of the derivative action. As this value increases, the derivative strength decreases.	0001 to 1FFF hex (1 to 8191); (0000 = Derivative operation not executed) (See note 1.)	
C+4	Sampling period (τ)	Sets the period for executing the PID action.	0001 to 270F hex (1 to 9999); (0.01 to 99.99 s, in units of 10 ms)	Not allowed
Bits 04 to 15 of C+5	2-PID parameter (α)	The input filter coefficient. Normally use 0.65 (i.e., a setting of 000). The filter efficiency decreases as the coefficient approaches 0.	000 hex: α = 0.65 Setting from 100 to 163 hex means that the value of the rightmost two digits is set from α= 0.00 to α= 0.99. (See note 2.)	
Bit 03 of C+5	Manipulated variable output designation	Designates the manipulated variable output for when the PV equals the SV.	0: Output 0% 1: Output 50%	
Bit 01 of C+5	PID constant change enable setting	The timing of enabling changes made to the proportional band (P), integral constant (Tik), and derivative constant (Tdk) for use in PID calculations.	0: At start of PID instruction execution 1: At start of PID instruction execution and each sampling period	Allowed

Control data	Item	Contents	Setting range	Change with ON input condition
Bit 00 of C+5	PID forward/reverse designation	Determines the direction of the proportional action.	0: Reverse action 1: Forward action	Not allowed
Bit 12 of C+6	Manipulated variable output limit control	Determines whether or not limit control will apply to the manipulated variable output.	0: Disabled (no limit control) 1: Enabled (limit control)	
Bits 08 to 11 of C+6	Input range	The number of input data bits.	0: 8 bits 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits7: 15 bits 3: 11 bits8: 16 bits 4: 12 bits	
Bits 04 to 07 of C+6	Integral and derivative unit	Determines the unit for expressing the integral and derivative constants.	1: Sampling period multiple 9: Time (unit: 100 ms)	
Bits 00 to 03 of C+6	Output range	The number of output data bits. (The number of output bits is automatically the same as the number of input bits.)	0: 8 bits 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits7: 15 bits 3: 11 bits8: 16 bits 4: 12 bits	
C +7	Manipulated variable output lower limit	The lower limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	
C +8	Manipulated variable output upper limit	The upper limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	
Bit 15 of C+9	AT Command Bit	This control bit starts autotuning. • Set the AT Command Bit to 1 to perform autotuning. (Autotuning can be started while PIDAT(191) is being executed.) • This bit is turned OFF automatically when autotuning is completed. Autotuning will be interrupted if the AT Command Bit is turned OFF manually. In this case, the PID constants will be enabled if they were already calculated when autotuning was interrupted.	As a Control Bit: • 0 → 1: Executes autotuning. • 1 → 0: Interrupts autotuning. (PID(191) turns the bit OFF automatically when autotuning is completed. As a Flag: 0: Autotuning is not being executed. 1: Autotuning is being executed.	Allowed

Control data	Item	Contents	Setting range	Change with ON input condition
Bits 00 to 11 of C+9	AT Calculation Gain	Set this parameter to adjust the contribution of the PID calculation results to the stored values. Normally, leave this parameter set to its default (0000). • Increase the value when emphasizing stability. • Decrease the value when emphasizing responsiveness.	0000 hex: 1.00 (Default) 0001 to 03E8 hex (1 to 1000); (0.01 to 10.00, in units of 0.01)	Allowed (These parameters are read when autotuning starts.)
C+10	Limit-cycle Hysteresis	Sets the hysteresis when the limit cycle is generated. The default setting for reverse operation turns ON the MV with a hysteresis of SV–20%. Increase this setting if a proper limit cycle cannot be generated because the PV is unstable. However, the AT accuracy will decline if the Limit-cycle Hysteresis is higher than necessary.	0000 hex: 0.20% (Default) 0001 to 03E8 hex: 0.01 to 10.00% in units of 0.01% FFFF hex: 0.00% **Note** The percentage is with respect to the input range.	

Note

(1) When the unit is designated as 1, the range is from 1 to 8,191 times the period. When the unit is designated as 9, the range is from 0.1 to 819.1 s. When 9 is designated, set the integral and derivative times to within a range of 1 to 8,191 times the sampling period.

(2) Setting the 2-PID parameter (α) to 000 yields 0.65, the normal value.

When the manipulated variable output limit control is enabled (i.e., set to "1"), set the values as follows:

0000 ≤ MV output lower limit ≤ MV output upper limit ≤ Max. value of output range

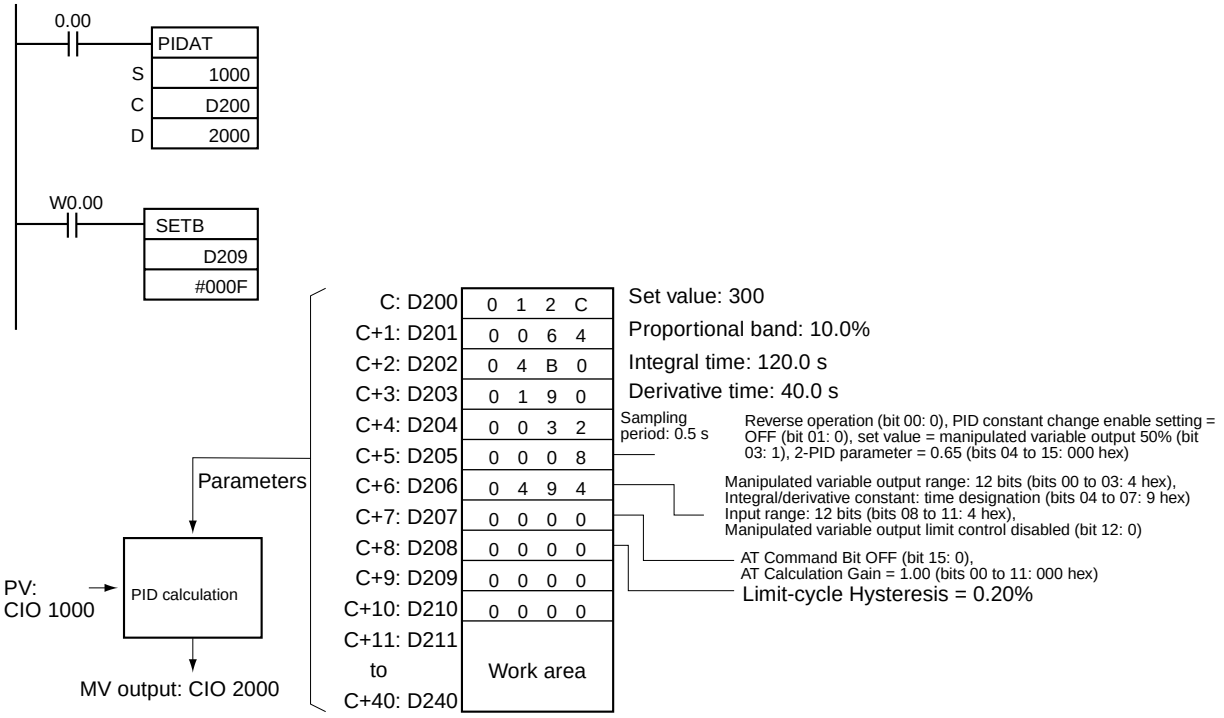
Example 1: Interrupting PID Control to Perform Autotuning

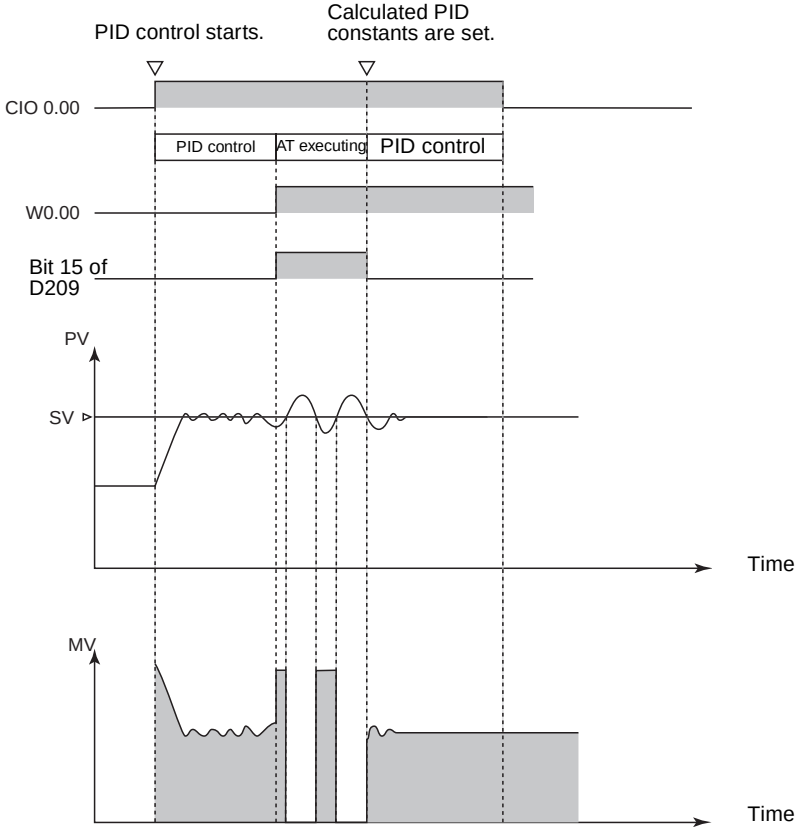
At the rising edge of CIO 0.00 (OFF to ON), the work area in D211 to D240 is initialized according to the parameters (shown below) set in D200 to D208. After the work area has been initialized, PID control is executed and the manipulated variable is output to CIO 2000.

While CIO 0.00 is ON, PID control is executed at the sampling period intervals according to the parameters set in D200 to D210. The manipulated variable is output to CIO 2000.

The PID constants used in PID calculations will not be changed even if the proportional band (P), integral constant (Tik), or derivative constant is changed after CIO 0.00 turns ON.

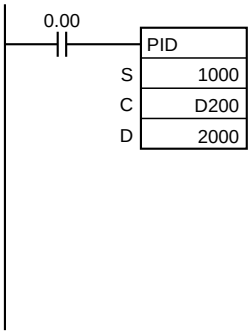
At the rising edge of W 0.00 (OFF to ON), SETB(532) turns ON bit 15 of D209 (C+9) and starts autotuning. When autotuning is completed, the calculated P, I, and D constants are written to C+1, C+2, and C+3. PID control is then restarted with the new PID constants.

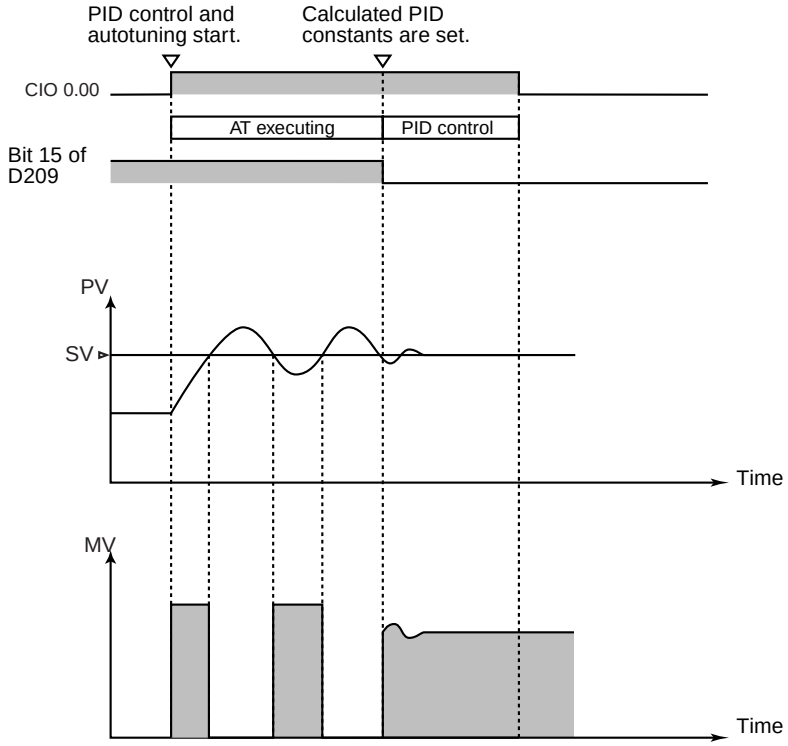

0.00
PIDAT
S 1000
C D200
D 2000
W0.00
SETB
D209
#000F
C: D200 0 1 2 C
C+1: D201 0 0 6 4
C+2: D202 0 4 B 0
C+3: D203 0 1 9 0
C+4: D204 0 0 3 2
C+5: D205 0 0 0 8
C+6: D206 0 4 9 4
C+7: D207 0 0 0 0
C+8: D208 0 0 0 0
C+9: D209 0 0 0 0
C+10: D210 0 0 0 0
C+11: D211
to
C+40: D240
Work area
Set value: 300
Proportional band: 10.0%
Integral time: 120.0 s
Derivative time: 40.0 s
Sampling period: 0.5 s
Reverse operation (bit 00: 0), PID constant change enable setting = OFF (bit 01: 0), set value = manipulated variable output 50% (bit 03: 1), 2-PID parameter = 0.65 (bits 04 to 15: 000 hex)
Manipulated variable output range: 12 bits (bits 00 to 03: 4 hex), Integral/derivative constant: time designation (bits 04 to 07: 9 hex) Input range: 12 bits (bits 08 to 11: 4 hex), Manipulated variable output limit control disabled (bit 12: 0)
AT Command Bit OFF (bit 15: 0), AT Calculation Gain = 1.00 (bits 00 to 11: 000 hex)
Limit-cycle Hysteresis = 0.20%
Parameters
PV: CIO 1000
PID calculation
MV output: CIO 2000


PID control starts.
Calculated PID constants are set.
CIO 0.00
PID control
AT executing
PID control
W0.00
Bit 15 of D209
PV
SV
Time
MV
Time

Example 2: Starting PIDAT(191) with Autotuning

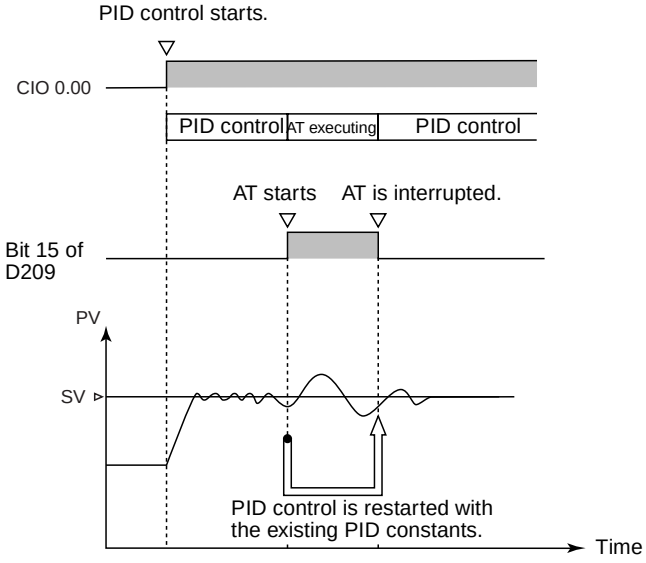
At the rising edge of CIO 0.00 (OFF to ON), autotuning will be performed first if bit 15 of D209 (C+9) is ON. When autotuning is completed, the calculated P, I, and D constants are written to C+1, C+2, and C+3. PID control is then started with the calculated PID constants.





Example 3: Interrupting Autotuning Before Completion

Autotuning can be interrupted by turning bit 15 of D209 (C+9) from ON to OFF. PID control will be restarted with the P, I, and D constants that were in effect before autotuning was started.

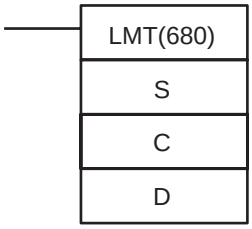


3-17-3 LIMIT CONTROL: LMT(680)

Purpose

Controls output data according to whether or not input data is within upper and lower limits.

Ladder Symbol



S: Input word

C: First limit word

D: Output word

Variations

Variations	**Executed Each Cycle for ON Condition**	LMT(680)
	Executed Once for Upward Differentiation	@LMT(680)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to 959	A0 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the execution condition is ON, LMT(680) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits. The contents of words C and C+1 are as follows:

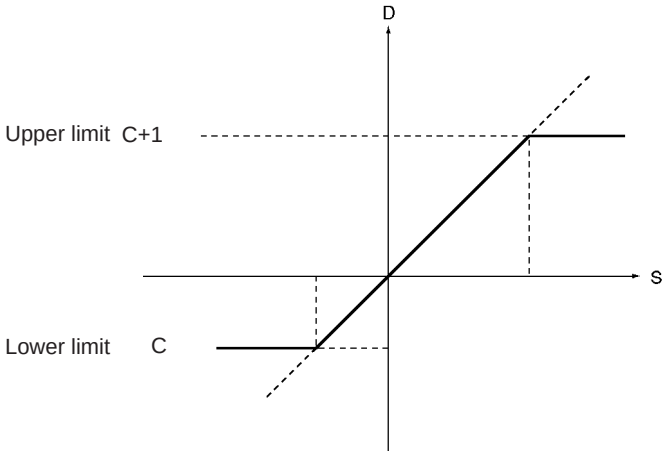
C	Lower limit data (minimum output data)
C+1	Upper limit data (maximum output data)

C and C+1 must have the same area classification.

If the input data (S) is less than the lower limit (C), the lower limit data will be output to D and the Less Than Flag will turn ON.

If the input data (S) is greater than the upper limit (C+1), the upper limit data will be output to D and the Greater Than Flag will turn ON.

If the input data (S) is greater than or equal to the lower limit (C) and less than or equal to the upper limit (C+1), the input data (S) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if the upper limit is less than the lower limit. OFF in all other cases.
Greater Than Flag	>	ON if the input data (S) is greater than the upper limit. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Less Than Flag	<	ON if the input data (S) is less than the lower limit. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the result is “1.” OFF in all other cases.

Precautions

If the upper limit is less than the lower limit, an error will occur and the Error Flag will turn ON.

If the input data (S) is greater than the upper limit, the Greater Than Flag will turn ON.

If the output word D is 0000 hex, the Equals Flag will turn ON.

If the input data (S) is less than the lower limit, the Less Than Flag will turn ON.

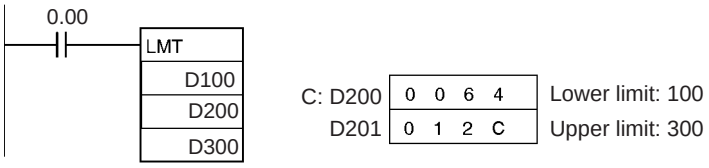
If the status of the leftmost bit of the output word D is “1,” the Negative Flag will turn ON.

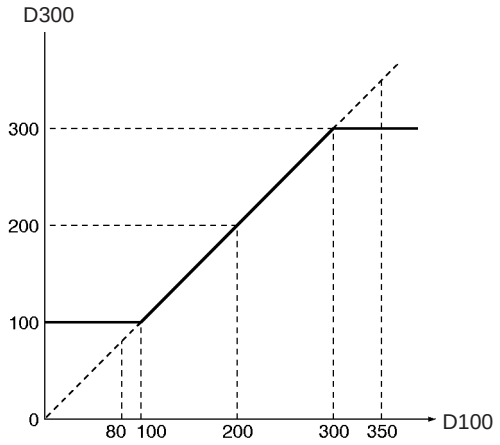
Example

If D100 is 0050 hex (80), then 0064 hex (100) will be output to D300 because 80 is less than the lower limit of 100.

If D100 is 00C8 hex (200), then 0064 hex (100) will be output to D300 because 200 is within the upper and lower limits.

If D100 is 012C hex (300), then 015E hex (350) will be output to D300 because 350 is greater than the upper limit of 300.



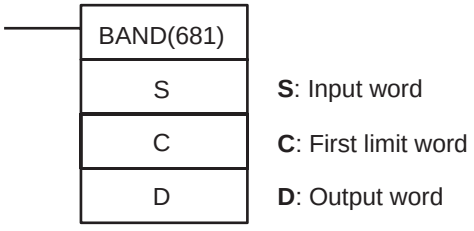


3-17-4 DEAD BAND CONTROL: BAND(681)

Purpose

Controls output data according to whether or not input data is within the lower and upper limits of the range (dead band range.)

Ladder Symbol



Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	BAND(681)
	Executed Once for Upward Differentiation	@BAND(681)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095

Area	S	C	D
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the execution condition is ON, BAND(681) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits (dead band). The contents of words C and C+1 are as follows:

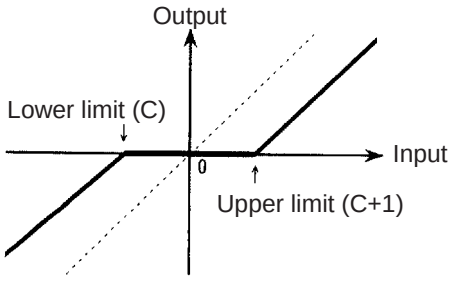
C	Lower limit data (dead band lower limit)
C+1	Upper limit data (dead band upper limit)

C and C+1 must have the same area classification.

If the input data (S) is greater than or equal to the lower limit (C) and less than or equal to the upper limit (C+1), 0000 (hex) will be output to D and the Equals Flag will turn ON.

If the input data (S) is less than the lower limit (C), the difference between the input data minus the lower limit data will be output to D and the Less Than Flag will turn ON.

If the input data (S) is greater than the upper limit (C+1), the difference between the input data minus the upper limit data will be output to D and the Greater Than Flag will turn ON.



If the output data is smaller than the 8000 (hex) or if is greater than 7FFF, the sign will be reversed. For example, for a lower limit of 0100 (hex) and input data of 8000 (hex), the output data will be as follows:
8000 (hex) [–32768] – 0100 (hex) [256] = 7F00 (hex) [32512]

Flags

Name	Label	Operation
Error Flag	ER	ON if the upper limit is less than the lower limit. OFF in all other cases.
Greater Than Flag	>	ON if the input data (S) is greater than the upper limit. OFF in all other cases.

Name	Label	Operation
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Less Than Flag	<	ON if the input data (S) is less than the lower limit. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the result is “1.” OFF in all other cases.

Precautions

If the upper limit is less than the lower limit, an error will occur and the Error Flag will turn ON.

If the input data (S) is greater than the upper limit, the Greater Than Flag will turn ON.

If the output word D is 0000 hex, the Equals Flag will turn ON.

If the input data (S) is less than the lower limit, the Less Than Flag will turn ON.

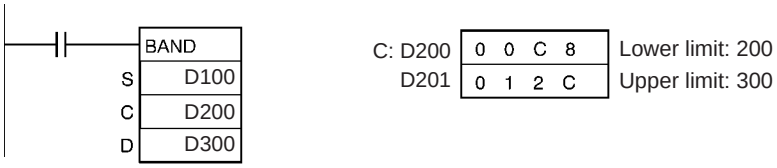
If the status of the leftmost bit of the output word D is “1,” the Negative Flag will turn ON.

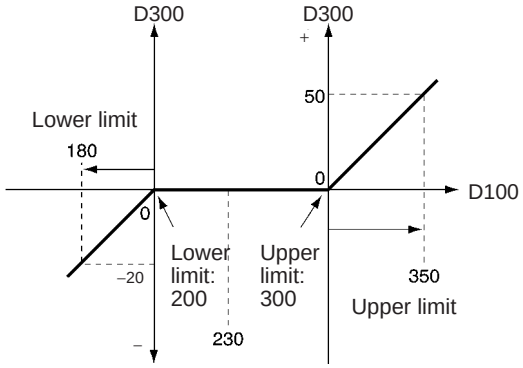
Example

If D100 is 00B4 hex (180), then 180–200=FFEC hex (–20) will be output to D300 because 180 is less than the lower limit of 200.

If D100 is 00E6 hex (230), then 0 will be output to D300 because 230 is within the upper and lower limits.

If D100 is 015E hex (350), then 350–300=0032 hex (50) will be output to D300 because 350 is greater than the upper limit of 300.



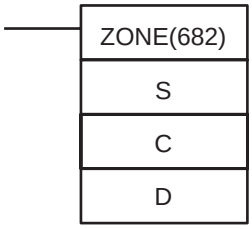


3-17-5 DEAD ZONE CONTROL: ZONE(682)

Purpose

Adds the specified bias to input data and outputs the result.

Ladder Symbol



S: Input word

C: First limit word

D: Output word

Variations

Variations		
	Executed Each Cycle for ON Condition	ZONE(682)
	Executed Once for Upward Differentiation	@ZONE(682)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

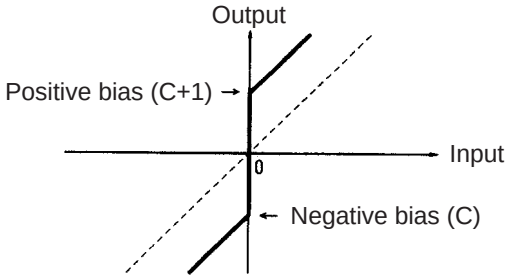
When the execution condition is ON, ZONE(682) adds the specified bias to the specified input data (signed 16-bit binary) and places the result in a specified word. The contents of words C and C+1 are as follows:

C	Negative bias
C+1	Positive bias

If the input data (S) is less than zero, the input data plus the negative bias will be output to D and the Less Than Flag will turn ON.

If the input data (S) is greater than zero, the input data plus the positive bias will be output to D and the Greater Than Flag will turn ON.

If the input data (S) is equal to zero, 0000 will be output to D and the Equals Flag will turn ON.



If the output data is smaller than the 8000 (hex) or if is greater than 7FFF, the sign will be reversed. For example, for a negative bias value of FF00 (hex) and input data of 8000 (hex), the output data will be as follows:
8000 (hex) [–32768] – FF00 (hex) [–256] = 7F00 (hex) [32512]

Flags

Name	Label	Operation
Error Flag	ER	ON if the upper limit is less than the lower limit. OFF in all other cases.
Greater Than Flag	>	ON if the input data (S) is greater than the upper limit. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Less Than Flag	<	ON if the input data (S) is less than the lower limit. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the result is “1.” OFF in all other cases.

Precautions

If the upper limit is less than the lower limit, an error will occur and the Error Flag will turn ON.

If the input data (S) is greater than the upper limit, the Greater Than Flag will turn ON.

If the output word D is 0000 hex, the Equals Flag will turn ON.

If the input data (S) is less than the lower limit, the Less Than Flag will turn ON.

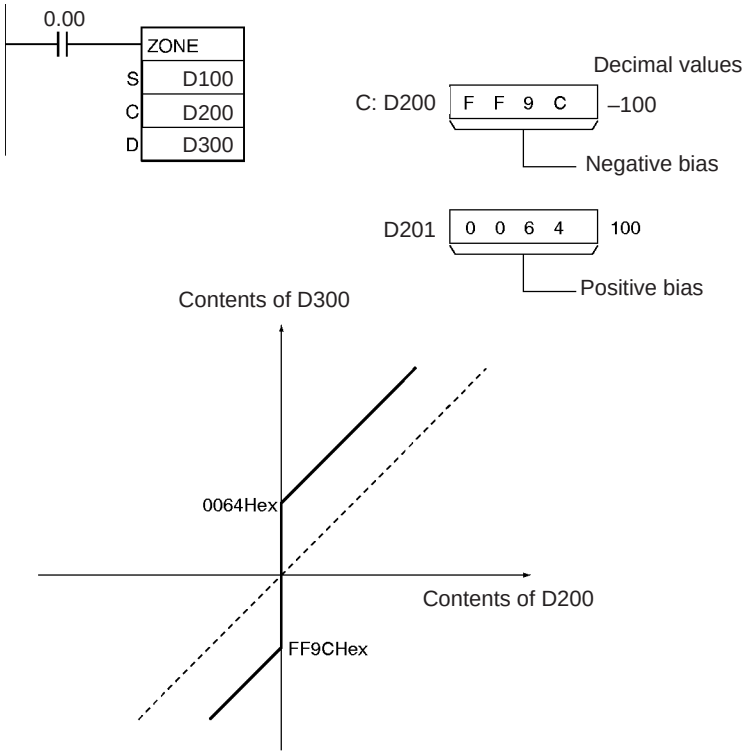
If the status of the leftmost bit of the output word D is “1,” the Negative Flag will turn ON.

Example

When CIO 0.00 is ON, a bias of –100 will be applied to the value of D100 if that value is less than 0, and the resulting value will be stored in D300.

If the value of D100 is 0, then 0000 hex will be stored in D300.

If the value of D100 is greater than 0, then a bias of +100 will be applied and the resulting value will be stored in D300.

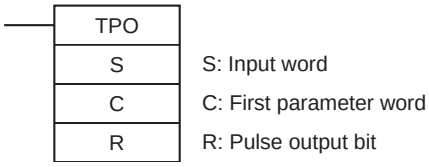


3-17-6 TIME-PROPORTIONAL OUTPUT: TPO(685)

Purpose

Inputs the duty ratio or manipulated variable from the specified word, converts the duty ratio to a time-proportional output based on the specified parameters, and outputs the result from the specified output.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	TPO(685)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

S: Input Word

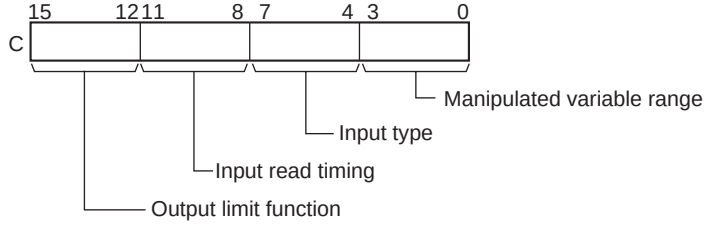
Specifies the input word containing the input duty ratio or manipulated variable. Bits 04 to 07 of C specify the input type, i.e., whether the input word contains an input duty ratio or manipulated variable. (Set these bits to 0 hex to specify a input duty ratio or to 1 hex to specify a manipulated variable.)

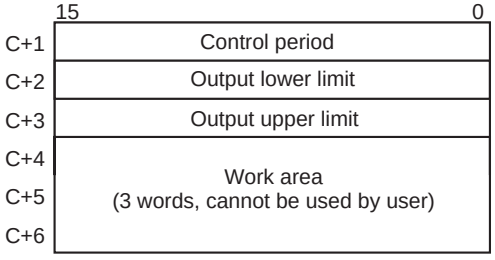
- Input duty ratio: 0000 to 2710 hex (0.00% to 100.00%)
- Input manipulated variable (See note.): 0000 to FFFF hex (0 to 65,535 max.) (Bits 00 to 03 of C specify the manipulated variable range, i.e., the number of valid bits in the manipulated variable. Specify the same number of bits as specified for the output range setting in PID(190).)

Note If S is a manipulated variable, specify the word containing the manipulated variable output from a PID(190) or PIDAT(191) instruction.

C to C+6: Parameters

The following diagram shows the locations of the parameter data. For details on the parameters, refer to *Parameter Settings* in this section.





Note: For details, see the description of each parameter.

R: Pulse Output Bit

Specifies the destination output bit for the pulse output.

Normally, specify an output bit allocated to a Transistor Output Unit and connect a solid state relay to the Transistor Output Unit.

Operand Specifications

Area	S	C	R
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6137	CIO 0.00 to CIO 6143.15
Work Area	W0 to W511	W0 to W505	W0.00 to W511.15
Holding Bit Area	H0 to H511	H0 to H505	H0.00 to H511.15
Auxiliary Bit Area	A0 to 959	A0 to A953	A448.00 to A959.15
Timer Area	T0000 to T4095	T0000 to T4089	---
Counter Area	C0000 to C4095	C0000 to C4089	---
DM Area	D0 to D32767	D0 to D32761	---
Indirect DM addresses in binary	@ D0 to @ D32767		---

Area	S	C	R
Indirect DM addresses in BCD	*D0 to *D32767		---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

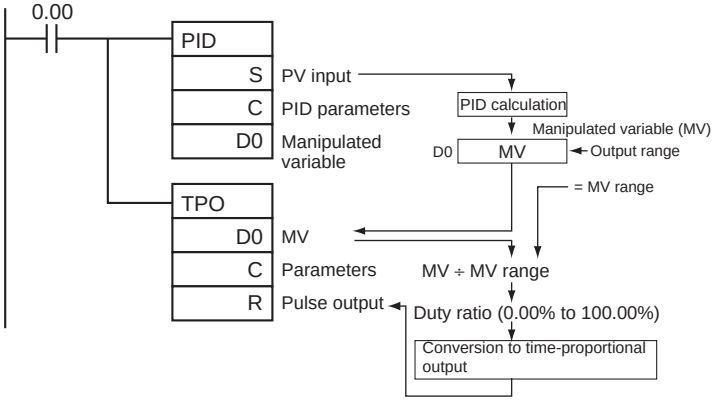
Receives a duty ratio or manipulated variable input from the word address specified by S, converts the duty ratio to a time-proportional output (see note) based on the parameters specified in words C to C+3, and outputs a pulse output to the bit specified by R.

Note A time-proportional output is changed proportionally based on the ON/OFF ratio in input word S. The period in which the ON and OFF status changes is known as the control period and is set in parameter word C+1.
Example: When the control period is 1 s and the input value is 50%, the bit is ON for 0.5 s and OFF for 0.5 s. When the control period is 1 s and the input value is 80%, the bit is ON for 0.8 s and OFF for 0.2 s.

Generally, TPO(685) is used together with PID(190) or PIDAT(191) and the PID instruction's manipulated variable result word (D) is specified as the input word (S) for the TPO(685) instruction. Also, an output bit allocated to a Transistor Output Unit is generally specified as R and a solid state relay is connected to the Transistor Output Unit to perform time-proportional control of a heater (proportional control of the ON/OFF ratio).

Combining TPO(685) with a PID Control Instruction

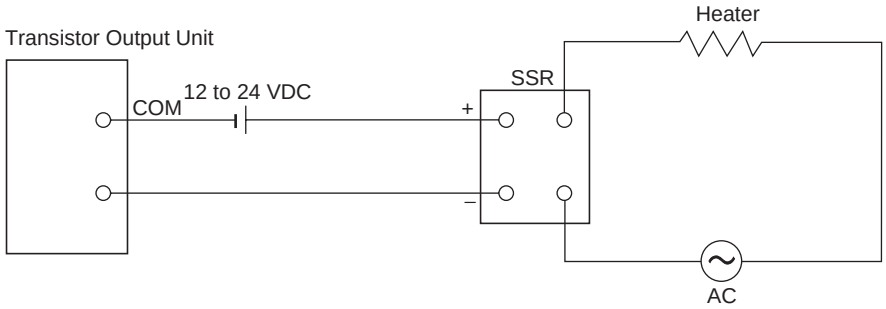
When combining TPO(685) with a PID control instruction, the manipulated variable input is divided by the manipulated variable range to calculate the duty ratio, that duty ratio is converted to a time-proportional output, and pulses are output.



In this case, set the same value for the PID Control instruction's output range and the TPO(685) instruction's manipulated variable range. For example, when the PID Control instruction's output range and the TPO(685) instruction's manipulated variable range are both set to 12 bits (0000 to 0FFF hex), the duty ratio is calculated by dividing the manipulated variable from the PID Control instruction by 0FFF hex and TPO(685) converts that duty ratio to a time-proportional output.

External Wiring Example

Connect the Transistor Output Unit to a solid state relay (SSR) as shown in the following diagram.



Parameter Settings

Control data		Item	Contents	Setting range	Change with ON input condition
Word	Bits				
C	00 to 03	Manipulated variable range	Specifies the number of input data bits.	0 hex: 8 bits 5 hex: 13 bits 1 hex: 9 bits 6 hex: 14 bits 2 hex: 10 bits 7 hex: 15 bits 3 hex: 11 bits 8 hex: 16 bits 4 hex: 12 bits	Allowed
	04 to 07	Input type	Specifies whether S contains a duty ratio or manipulated variable.	0 hex: Duty ratio Setting range for S: 0000 to 2710 hex (0.00 to 100.00%) 1 hex: Manipulated variable Setting range for S: 0000 to FFFF hex (0 to 65,535) (The maximum setting depends on the MV range set with bits 00 to 03 of C.)	Allowed
	08 to 11	Input read timing	Specifies the input read timing.	0 hex: Use the beginning value of the control period 1 hex: Use lower value 2 hex: Use higher value 3 hex: Continuous adjustment	Allowed
	12 to 15	Output limit control	Specifies whether the output limit function is enabled or disabled.	0 hex: Disabled 1 hex: Enabled (See note.)	Allowed
C+1	00 to 15	Control period	Control period (Time period in which the ON/OFF changes are made.)	0064 to 270F hex (1.00 to 99.99 s) **Note** For example, 1.00 s is set as 0064 hex, and not 0001 hex.	Allowed
C +2	00 to 15	Output lower limit	Specifies the lower limit when the output limit is enabled.	0000 to 2710 hex (0 to 100.00%)	Allowed
C +3	00 to 15	Output upper limit	Specifies the upper limit when the output limit is enabled.	0000 to 2710 hex (0 to 100.00%)	Allowed
C+4	00 to 15	Work area	This work area is used by the system. It cannot be used by the user.	Cannot be used.	---
C+5	00 to 15				
C+6	00 to 15				

Note When the output limit control function is enabled, set the lower and upper limits as follows: 0000 hex ≤ lower limit ≤ upper limit ≤ 2710 hex.

Execution

- The instruction is executed while the input condition is ON.
- When instruction execution starts, the output bit (R) is turned ON/OFF according to the duty ratio.
- The parameters (in C to C+3) are read in real time each time that the instruction is executed. When changing the parameters, change all of them at the same time so that different sets of parameters are not mixed.
- The output (R) is turned ON/OFF when the instruction is executed and the accuracy of the output's ON/OFF timing is 10 ms max.
- Execution of the instruction stops when the input condition goes OFF. At that time, the elapsed time value will be reset and the control period will be initialized.
- The input type setting (bits 04 to 07 of C) determines whether the input word (S) contains a duty ratio or manipulated variable. When S contains the manipulated variable, the duty ratio is calculated by dividing the manipulated variable input by the manipulated variable range (bits 00 to 03 of C).

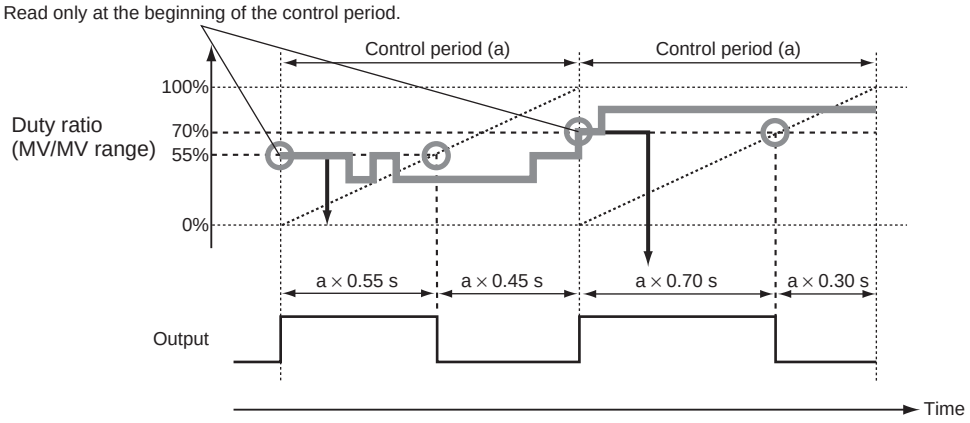
Input Read Timing Setting (C bits 08 to 11)

The input read timing setting (bits 08 to 11 of C) specifies when the input word (S) is read, as shown in the following table:

Input read timing	Description
0: Use the beginning value of the control period	The duty ratio input is read at the beginning of the control period and the ratio cannot be changed during the control period.
1: Use lower value	If the duty ratio input falls below the duty ratio at the beginning of the control period, the lower value will take precedence and the output ON time will be reduced accordingly.
2: Use higher value	If the duty ratio input rises above the duty ratio at the beginning of the control period, the higher value will take precedence and the output ON time will be increased accordingly.
3: Continuous adjustment	The duty ratio will be read in real time each time the instruction is executed and the ON/OFF operation will be repeated within the control period.

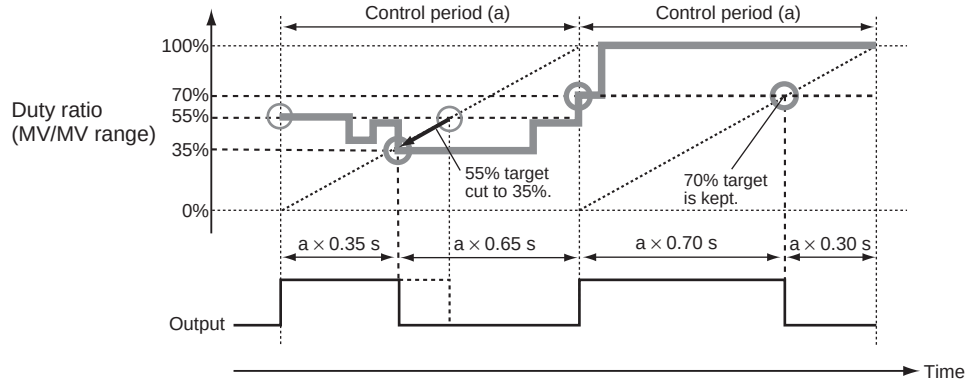
The following diagrams show the operation of each input read timing setting.

- Input time setting = 0 (Use the beginning value of the control period.)



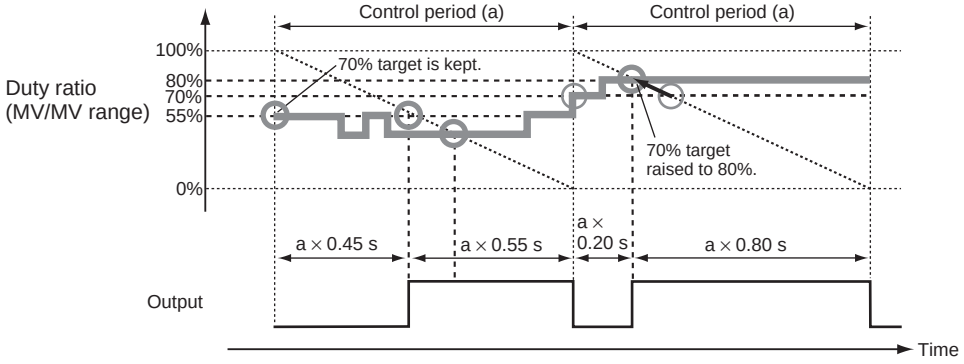
Each control period's output is determined by the duty ratio at the beginning of that period.
Use this setting for general applications.

- Input time setting = 1 (Use lower value.)



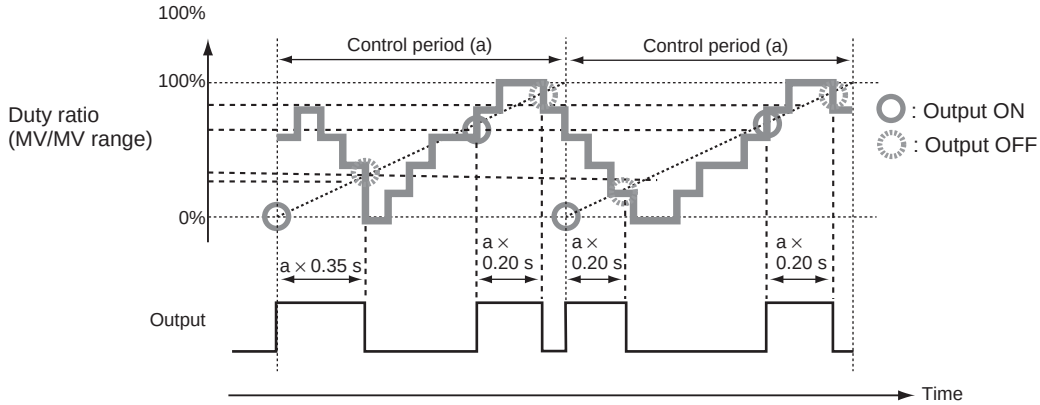
If the duty ratio falls below the initial value early enough, the duty ratio will be adjusted and the output will be turned OFF sooner.
Use this setting for applications such as avoiding overshooting when using time-proportional control to control heating and using a relatively long control period.

• Input time setting = 2 (Use higher value.)



If the duty ratio rises above the initial value early enough, the duty ratio will be adjusted and the output will be turned ON sooner. (With this setting the output's ON/OFF order is reversed and the output goes from OFF to ON.)
Use this setting for applications such as avoiding undershooting when using time-proportional control to control cooling and using relatively long control period.

• Input time setting = 3 (Continuous adjustment)



Changes in the duty ratio are monitored in real time. If the duty ratio falls below the initial value early enough, the duty ratio will be adjusted and the output will be turned OFF sooner. If the duty ratio rises again after that, the ratio will be adjusted again and the output will be turned ON. This process is repeated continuously.
Use this setting to improve responsiveness when the control period is relatively long and the duty ratio changes quickly. This setting is also appropriate for lighting or power applications that require precise control.

Flags

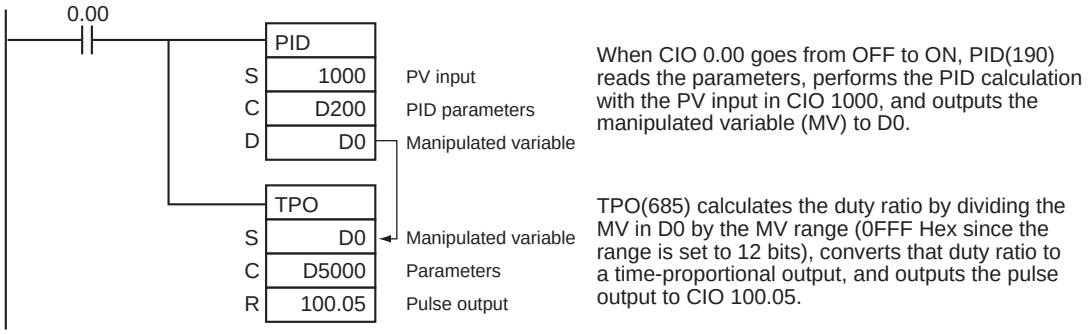
Name	Label	Operation
Error Flag	ER	ON if the input data in S is out of range. (The input data setting range depends on the input type setting.) ON if the C data is out of range. (The manipulated variable range will cause an error only when the input type is set to manipulated variable.) ON if the control period in C+1 is out of range. ON if the output limit function is enabled but the output lower limit (C+2) or output upper limit (C+3) is out of range. ON if the output limit function is enabled but the output lower limit (C+2) is less than or equal to the output upper limit (C+3). OFF in all other cases.

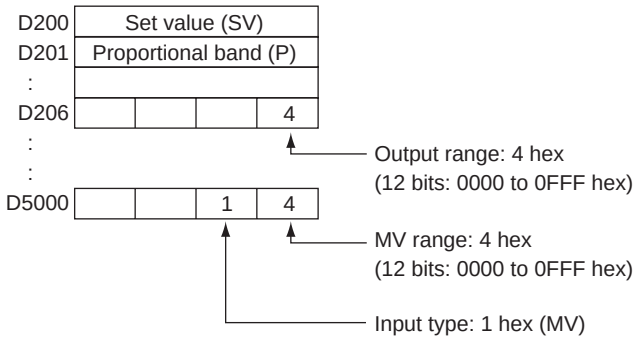
Example

Example 1: Combining TPO(685) with PID(190)

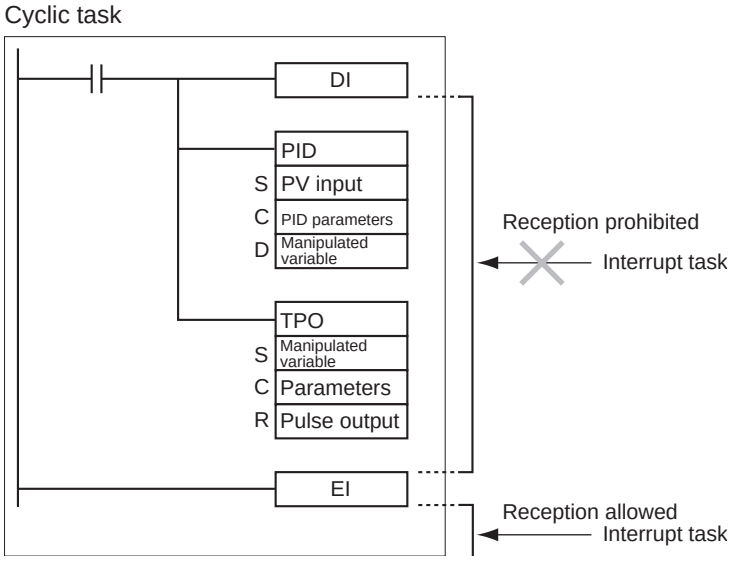
When CIO 0.00 is ON, TPO(685) takes the manipulated variable output from PID(190) (contained in D0), calculates the duty ratio from that manipulated variable value (Duty ratio = MV ÷ MV range), converts the duty ratio to a time-proportional output, and outputs the pulses to CIO 100.05.

In this case, CIO 100 is allocated to a Transistor Output Unit and bit CIO 100.05 is connected to a solid state relay for heater control.





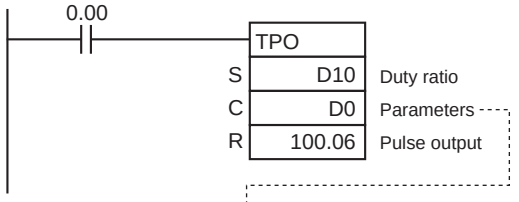
Note When using TPO(685) in combination with PID(190) in a cyclic task and also using an interrupt task, temporarily disable interrupts by executing DI(693) (DISABLE INTERRUPTS) ahead PID(190) and TPO(685). If interrupts are not disabled and an interrupt occurs between the PID(190) and TPO(685), the control period may be shifted.



Example 2: Using TPO(685) Alone

When CIO 0.00 is ON, TPO(685) takes the duty ratio in D10, converts the duty ratio to a time-proportional output, and outputs the pulses to CIO 100.06.

In this case, the control period is 1 s and the output limit function is enabled with a lower limit 20.00% and an upper limit of 80.00%.



TPO(685) takes the duty ratio in D10, converts that duty ratio to a time-proportional output, and outputs the pulse output to bit 00 of CIO 100.06.

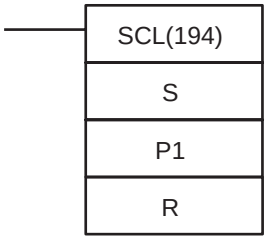
D0	1	1	0	0	Duty ratio input, read initial value, and enable output limit function.
D1	0	0	6	4	Control period = 1.00 s
D2	0	7	D	0	Output lower limit = 20.00%
D3	1	F	4	0	Output upper limit = 80.00%
D4	Do not set.				
D5	Do not set.				
D6	Do not set.				
:					
:					
D10	0 to 2710 hex				0 to 100.00%

3-17-7 SCALING: SCL(194)

Purpose

Converts unsigned binary data into unsigned BCD data according to the specified linear function.

Ladder Symbol



S: Source word

P1: First parameter word

R: Result word

Variations

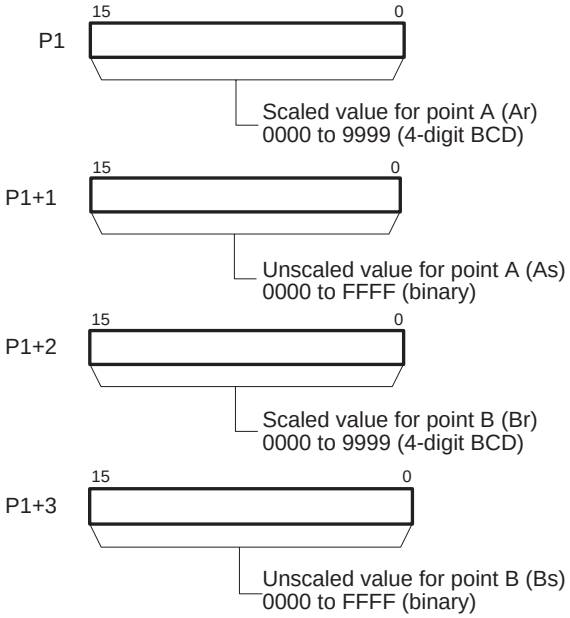
Variations		
	Executed Each Cycle for ON Condition	SCL(194)
	Executed Once for Upward Differentiation	@SCL(194)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

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.



Operand Specifications

Area	S	P1	R
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6140	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W508	W0 to W511
Holding Bit Area	H0 to H511	H0 to H508	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A956	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4092	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4092	C0000 to C4095
DM Area	D0 to D32767	D0 to D32764	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	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 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, Ad) and (Bs, Bd). 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 define by 2 values (As and Bs) before scaling and 2 values (Ar and Br) after scaling.

The following equations are used for the conversion.

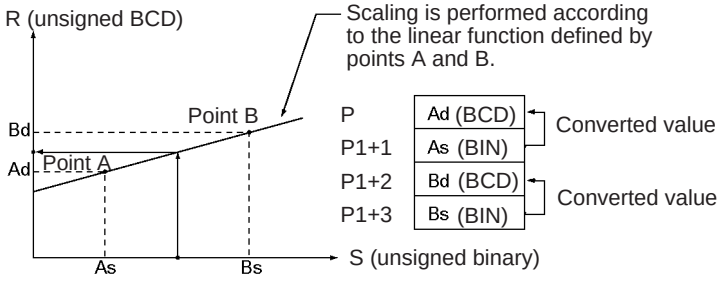
$$R = Bd - \frac{(Bd - Ad)}{\text{BCD conversion of } (Bs - As)} \times \text{BCD conversion of } (Bs - S)$$

The slope of the line is as follows:

$$R = Bd - \frac{(Bd - Ad)}{\text{BCD conversion of } (Bs - As)}$$

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 Analog Input Units according to user-defined scale parameters. For example, if a 1 to 5-V input to an Analog Input Unit is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to 50 to 200°C 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 C (Ar) or C+1 (Br) is not BCD. ON if the contents of C+1 (As) and C+3 (Bs) are equal. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.

Precautions

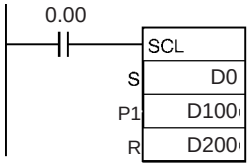
An error will occur and the Error Flag will turn ON if the values for Ar (C) and Br (C+2) are not in BCD, or if the values for As (C+1) and Bs (C+3) are equal.

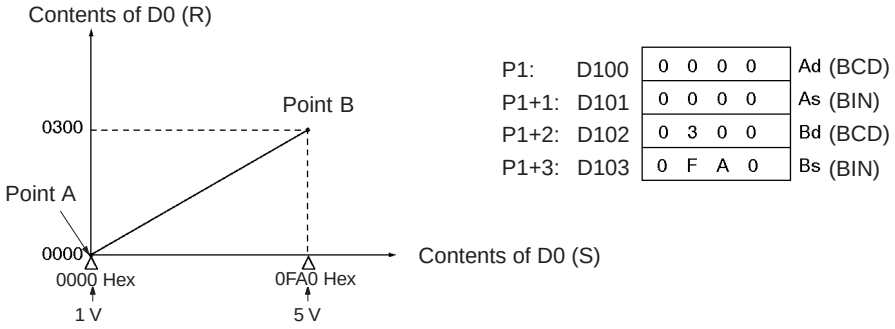
The Equals Flag will turn ON when the contents of the result word D is 0000.

Examples

In the following example, it is assume that an analog signal from 1 to 5 V is converted and input to D0 as 0000 to 0FA0 hexadecimal. SCL(194) is used to convert (scale) the value in CIO 200 to a value between 0000 and 0300 BCD.

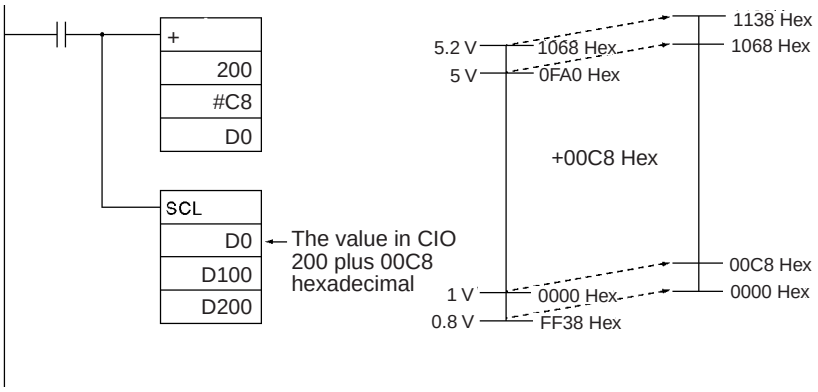
When CIO 0.00 is ON, the contents of D0 is scaled using the linear function defined by point A (0000, 0000) and point B (0FA0, 0300). The coordinates of these points are contained in D100 to D103, and the result is output to D200.

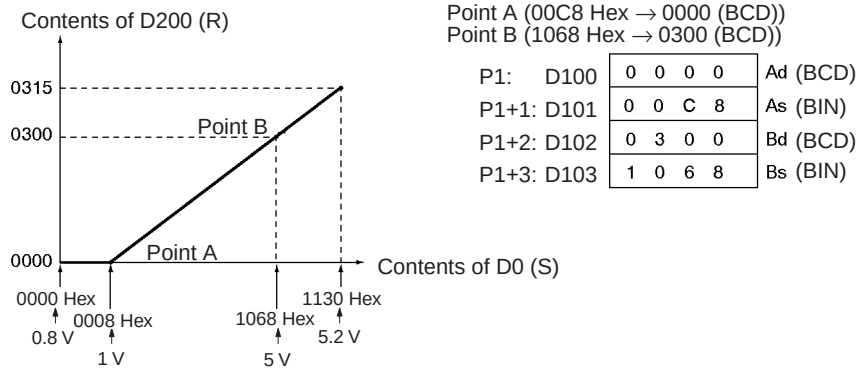




Negative Values

An Analog Input Unit 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.

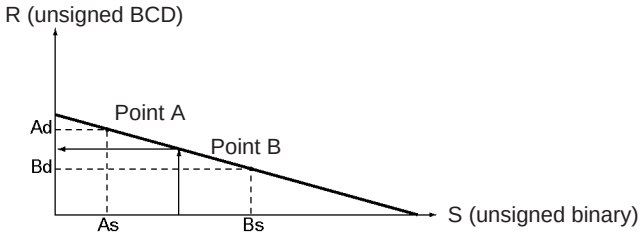




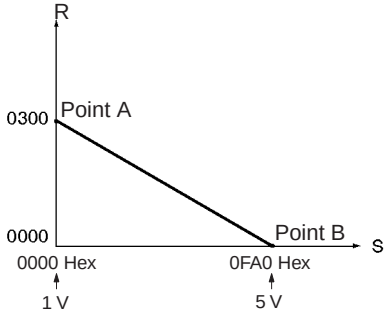
In this example, values from 0000 to 00C8 hexadecimal will be converted to negative values. SCL(194), however, can output only unsigned BCD values from 0000 to 9999, so 0000 BCD will be output whenever the contents of D0 is between 0000 and 00C8 hexadecimal.

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.

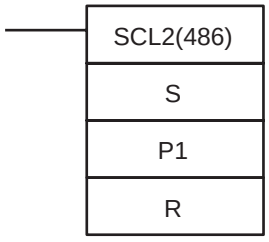


3-17-8 SCALING 2: SCL2(486)

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

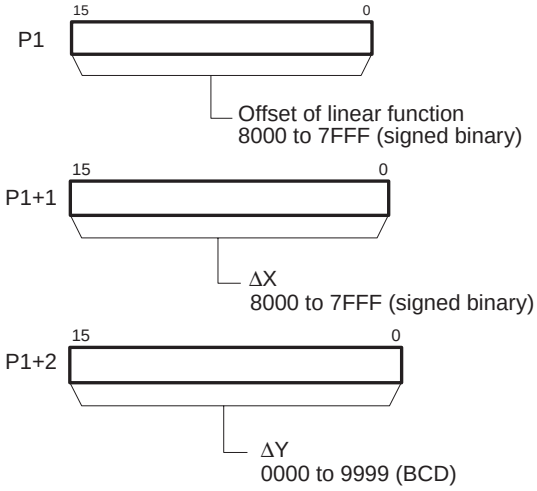
Variations	**Executed Each Cycle for ON Condition**	SCL2(486)
	Executed Once for Upward Differentiation	@SCL2(486)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

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.a



Operand Specifications

Area	S	P1	R
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6141	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W509	W0 to W511
Holding Bit Area	H0 to H511	H0 to H509	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A957	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4093	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4093	C0000 to C4095
DM Area	D0 to D32767	D0 to D32765	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	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 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.

$$R = \frac{\Delta Y}{\text{BCD conversion of } \Delta X} \times ((\text{BCD conversion of S}) - (\text{BCD conversion of offset})$$

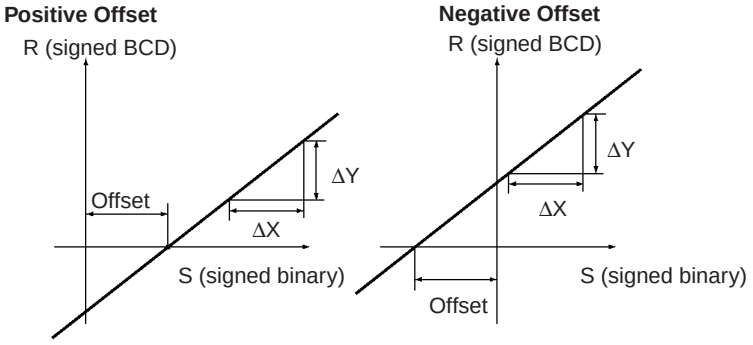
The slope of the line is ΔY/ΔX.

The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

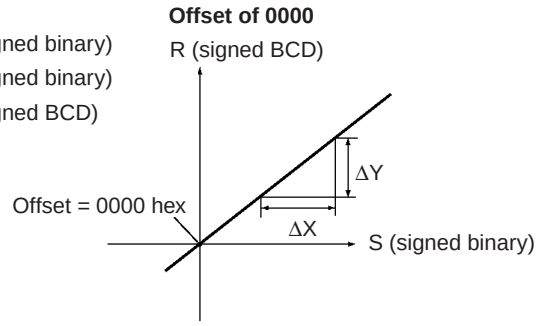
The result will be rounded to the nearest integer.

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.



P1	Offset	(Signed binary)
P1+1	ΔY	(Signed binary)
P1+2	ΔX	(Signed BCD)



SCL2(486) can be used to scale the results of analog signal conversion values from Analog Input Units according to user-defined scale parameters. For example, if a 1 to 5-V input to an Analog Input Unit is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to –100 to 200°C 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 C+1 (ΔX) is 0000. ON if the contents of C+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 (C+1) is 0000 or if the value for ΔY (C+2) is not BCD.

The Equals Flag will turn ON when the contents of the result word D is 0000.

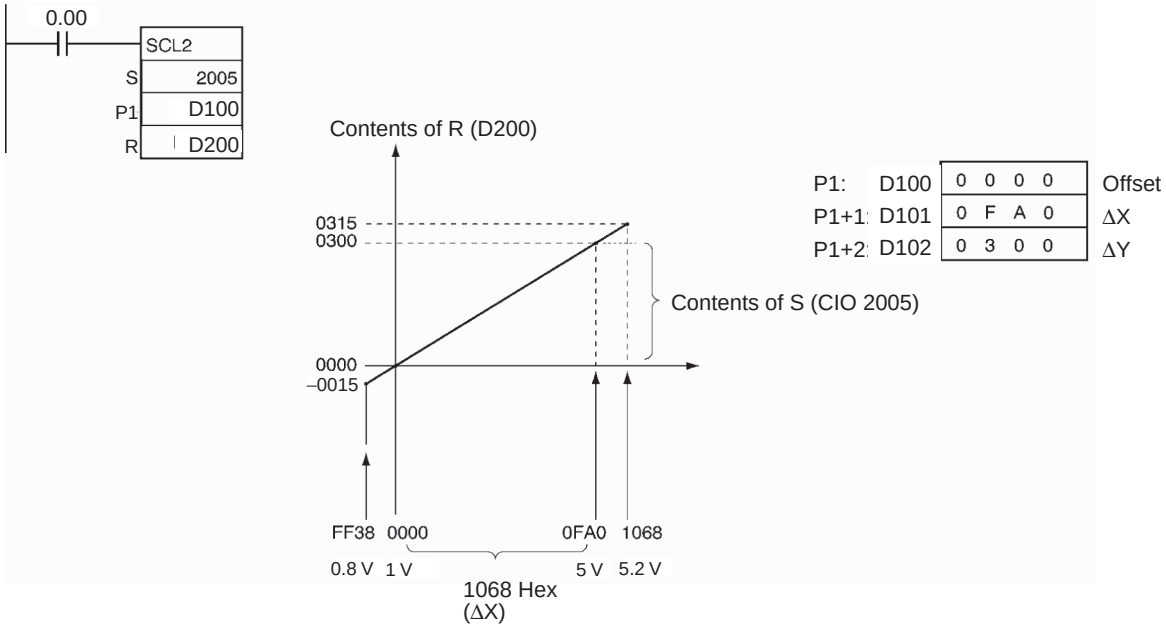
The Carry Flag will turn ON if the value placed in the result word is negative.

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 2005 as 0000 to 0FA0 hexadecimal. SCL2(486) is used to convert (scale) the value in CIO 2005 to a value between 0000 and 0300 BCD.

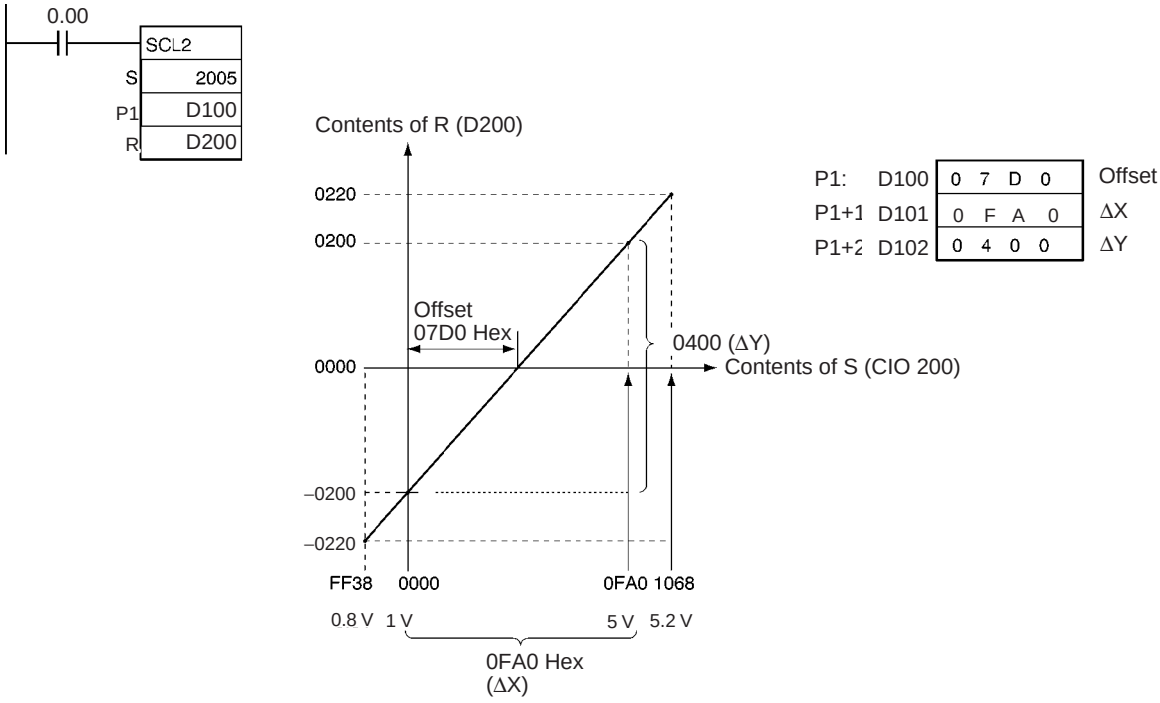
When CIO 0.00 is ON, the contents of CIO 2005 is scaled using the linear function defined by ΔX (0FA0), ΔY (0300), and the offset (0). These values are contained in D100 to D102, and the result is output to D200.



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 2005 as 0000 to 0FA0 hexadecimal. SCL2(486) is used to convert (scale) the value in CIO 2005 to a value between –0200 and 0200 BCD.

When CIO 0.00 is ON, the contents of CIO 2005 is scaled using the linear function defined by ΔX (0FA0), ΔY (0400), and the offset (07D0). These values are contained in D100 to D102, and the result is output to D200.

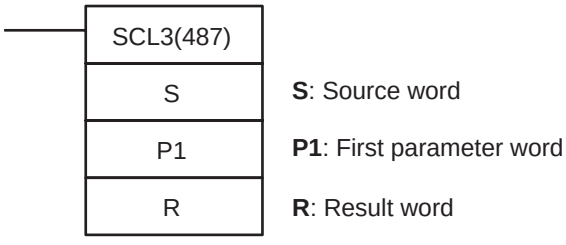


3-17-9 SCALING 3: SCL3(487)

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



S: Source word

P1: First parameter word

R: Result word

Variations

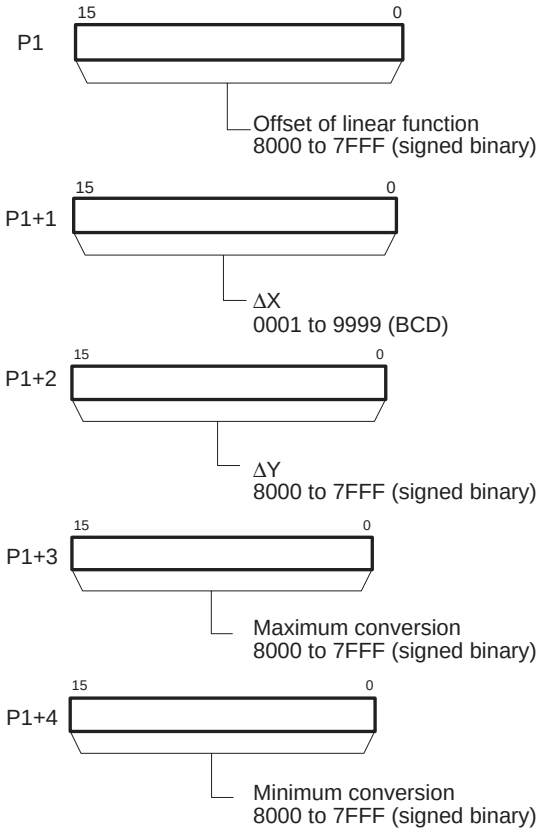
Variations		
	Executed Each Cycle for ON Condition	SCL3(487)
	Executed Once for Upward Differentiation	@SCL3(487)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

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.



Operand Specifications

Area	S	P1	R
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6139	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W507	W0 to W511
Holding Bit Area	H0 to H511	H0 to H507	H0 to H511
Auxiliary Bit Area	A0 to A447 A448 to A959	A0 to A443 A448 to A955	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4091	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4091	C0000 to C4095
DM Area	D0 to D32767	D0 to D32763	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	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 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 ΔY/ΔX.

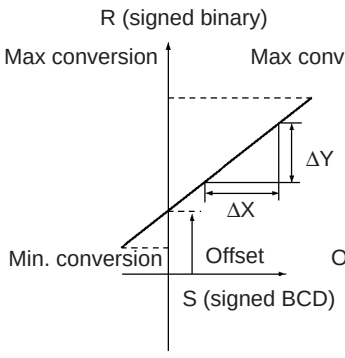
The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer.

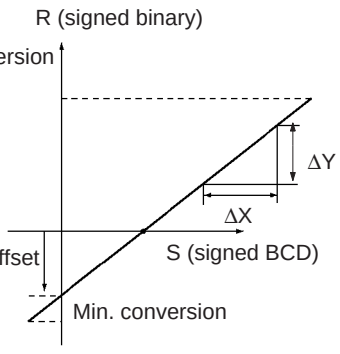
The source value in S is treated as an absolute BCD value and the sign is indicated by the Carry Flag. The source value can thus be between –9999 and 9999.

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.

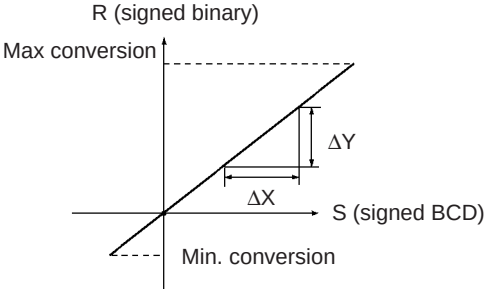
Positive Offset



Negative Offset



Offset of 0000



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 C+1 (ΔX) is not between 0001 and 9999 BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0. 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 (C+1) is not between 0001 and 9999 BCD.

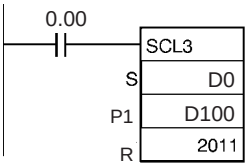
The Equals Flag will turn ON when the contents of the result word D is 0000.

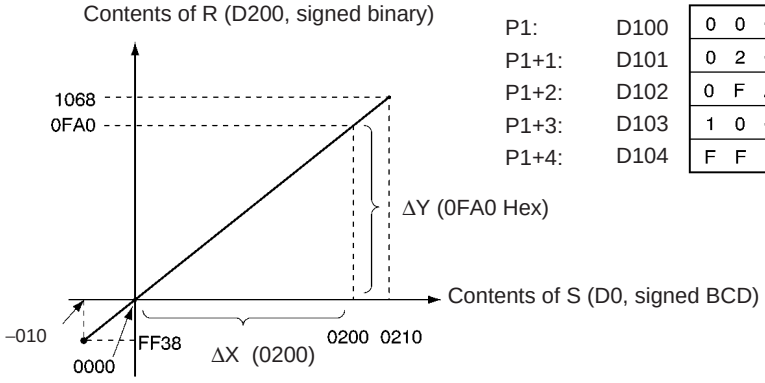
The Negative Flag will turn ON if the MSB of the result in R is 1, i.e., if the result is negative.

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 an Analog Output Unit.

When CIO 0.00 turns ON in the following example, the contents of D0 is scaled using the linear function defined by ΔX (0200), ΔY (0FA0), and the offset (0). These values are contained in D100 to D102. The sign of the BCD value in D0 is indicated by the Carry Flag. The result is output to CIO 2011.





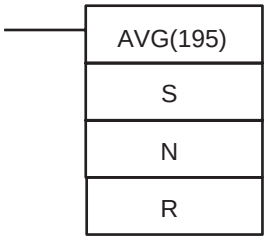
P1:	D100	0 0 0 0	Offset
P1+1:	D101	0 2 0 0	ΔX
P1+2:	D102	0 F A 0	ΔY
P1+3:	D103	1 0 6 8	Max. conversion
P1+4:	D104	F F 3 8	Min. conversion

3-17-10 AVERAGE: AVG(195)

Purpose

Calculates the average value of an input word for the specified number of cycles.

Ladder Symbol



S: Source word

N: Number of cycles

R: Result word

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

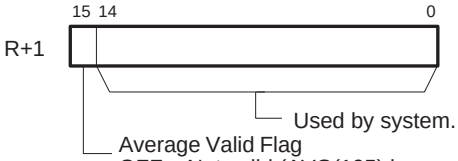
N: Number of Cycles

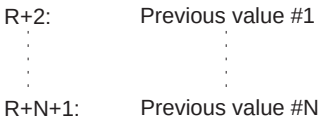
The number of cycles must be between 0001 and 0040 hexadecimal (0 to 64 cycles).

R: Result Word and R+1: First Work Area Word

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+1 contain the previous values of S as shown in the following diagram.

R: Average
R+1: Processing information





Operand Specifications

Area	S	N	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		

Area	S	N	R
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	#0001 to #0040 (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 IR15		

Description

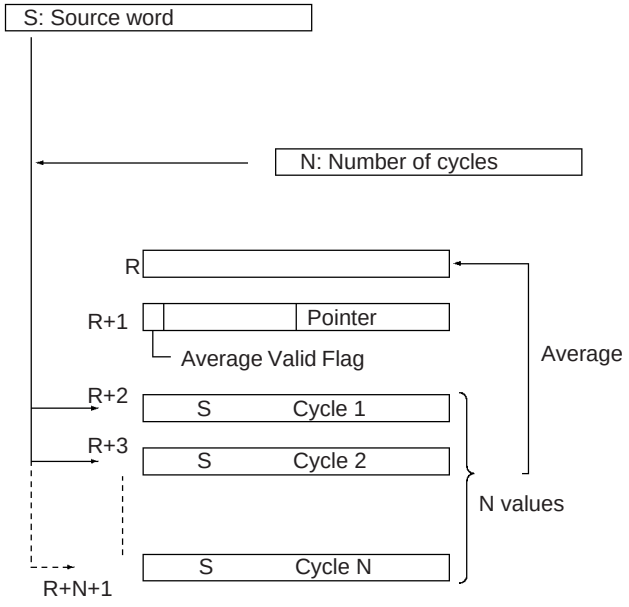
For the first N–1 cycles when the execution condition is ON, AVG(195) writes the values of S in order to words starting with R+2. The Previous Value Pointer (bits 00 to 07 of R+1) is incremented each time a value is written. Until the Nth value is written, the contents of S will be output unchanged to R and the Average Value Flag (bit 15 of R+1) will remain OFF.

When the Nth value is written to R+N+1, the average of all the values that have been stored will be computed, the average will be output to R as an 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 contents of N is 0. OFF in all other cases.

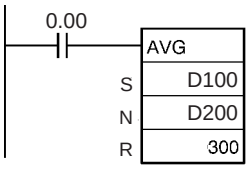
Precautions

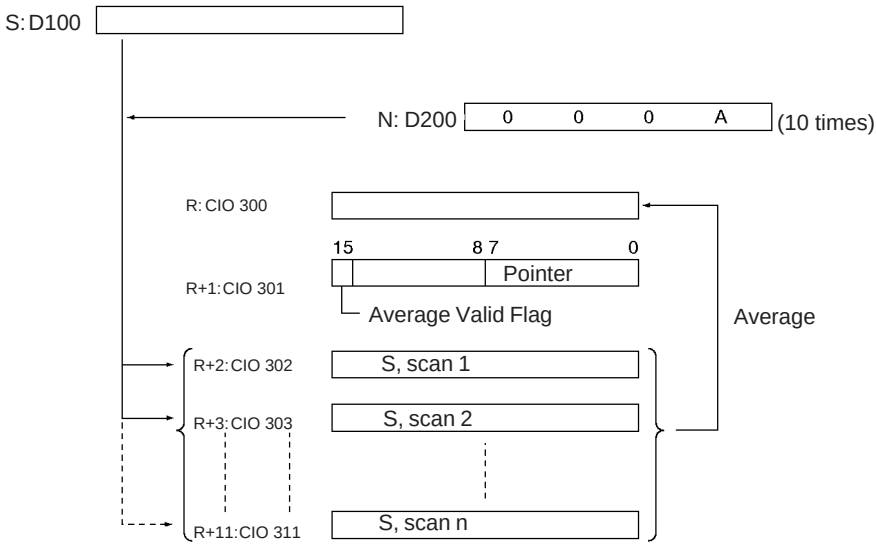
The contents of the First Work Area Word (D+1) is cleared to 0000 each time the execution condition changes from OFF to ON.

The contents of the First Work Area Word (D+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) contains 0000, an error will occur and the Error Flag will turn ON.

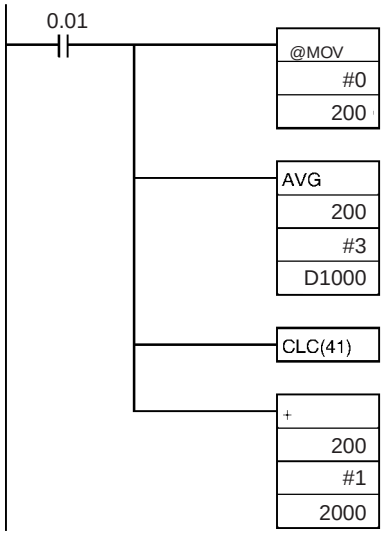
When CIO 0.00 is ON in the following example, the contents of D100 will be stored one time each scan for the number of scans specified in D200. The contents will be stored in order in the ten words from CIO 302 to CIO 311. The average of the contents of these ten words will be placed in CIO 300 and then bit 15 of CIO 301 will be turned ON.





Examples

In the following example, the content of CIO 200 is set to #0000 and then incremented by 1 each cycle. For the first two cycles, AVG(195) moves the content of CIO 200 to D1002 and D1003. The contents of D1001 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 D1002 to D1004 and writes that average value to D1000.



	1st cycle	2nd cycle	3rd cycle	4th cycle
CIO 200	0000	0001	0002	0003

D1000	0000	0001	0001	0002	Average
D1001	0001	0002	8000	8001	Pointer
D1002	0000	0000	0000	0003	3 previous values of IR 40
D1003	---	0001	0001	0001	
D1004	---	---	0002	0002	

3-18 Subroutines

This section describes instructions used to create and control subroutines.

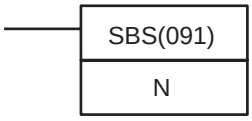
Instruction	Mnemonic	Function code	Page
SUBROUTINE CALL	SBS	091	669
MACRO	MCRO	099	675
SUBROUTINE ENTRY	SBN	092	679
SUBROUTINE RETURN	RET	093	681
GLOBAL SUBROUTINE CALL	GSBS	750	682
GLOBAL SUBROUTINE ENTRY	GSBN	751	689
GLOBAL SUBROUTINE RETURN	GRET	752	692

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		
Variations	**Executed Each Cycle for ON Condition**	SBS(091)
	Executed Once for Upward Differentiation	@SBS(091)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Subroutine number

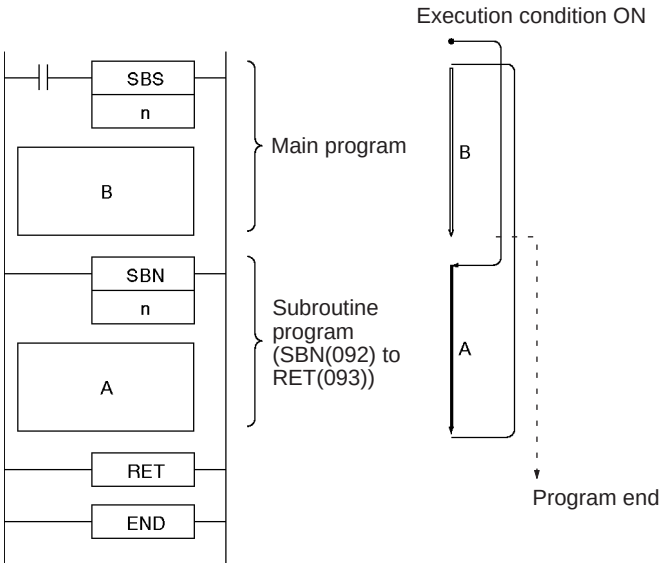
Specifies the subroutine number between 0 and 255 decimal.

Operand Specifications

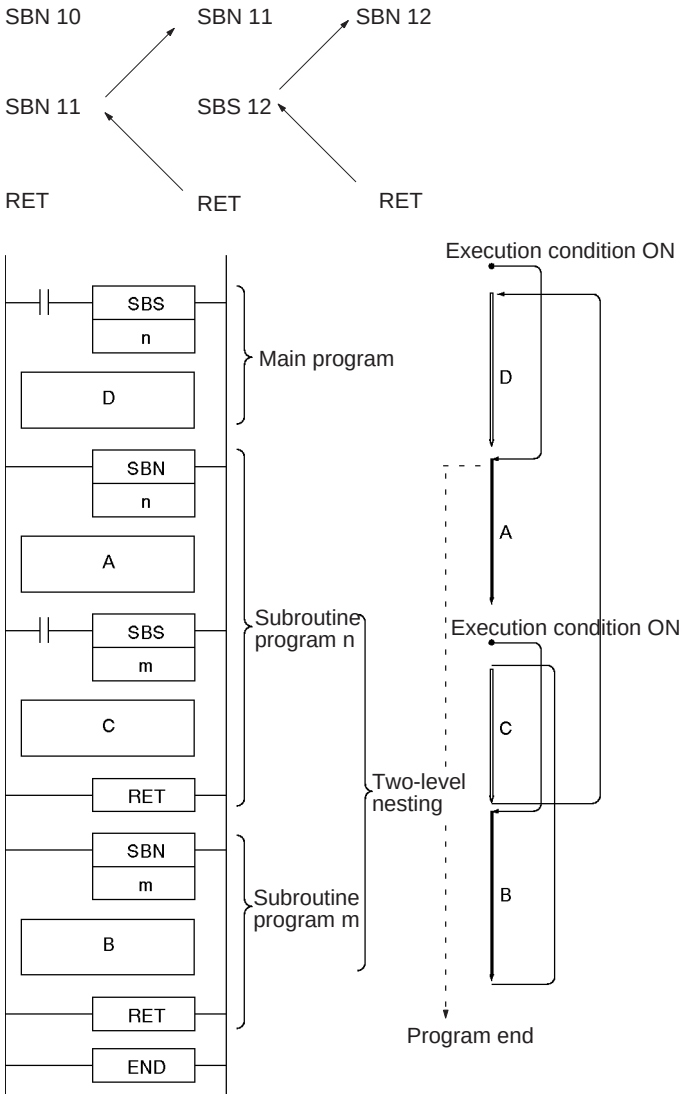
Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	0 to 255 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

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.

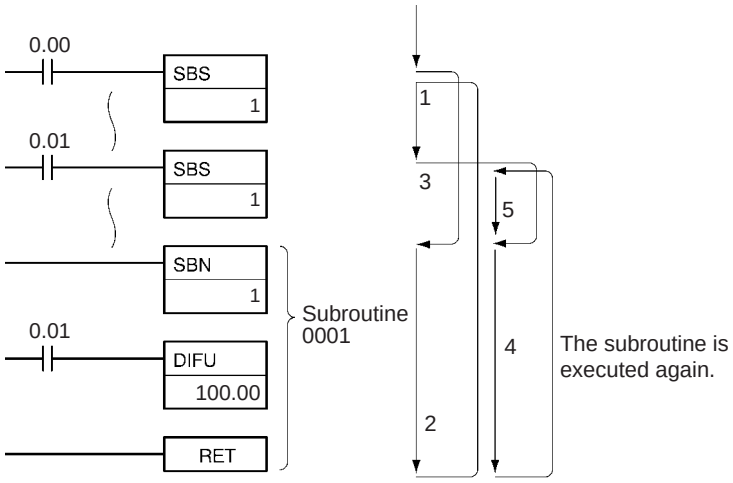


Note A subroutine can be called more than once in a program.

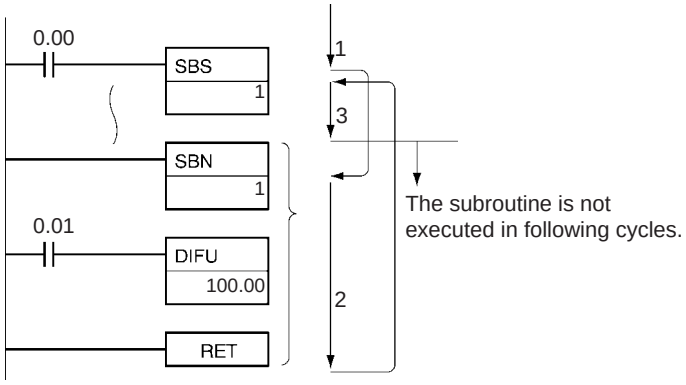
Subroutines and Differentiation

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 0001 is executed when CIO 0.00 is ON and CIO 100.00 is turned ON by DIFU(013) when CIO 0.01 has gone from OFF to ON. If CIO 0.01 is ON in the same cycle, subroutine 0001 will be executed again but this time DIFU(013) will turn CIO 100.00 OFF without checking the status of CIO 0.01.



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 0001 is executed if CIO 0.00 is ON. Output CIO 100.00 is turned ON by DIFU(013) when CIO 0.01 has gone from OFF to ON. If CIO 0.00 is OFF in the following cycle, subroutine 0001 will not be executed again and output CIO 100.00 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:

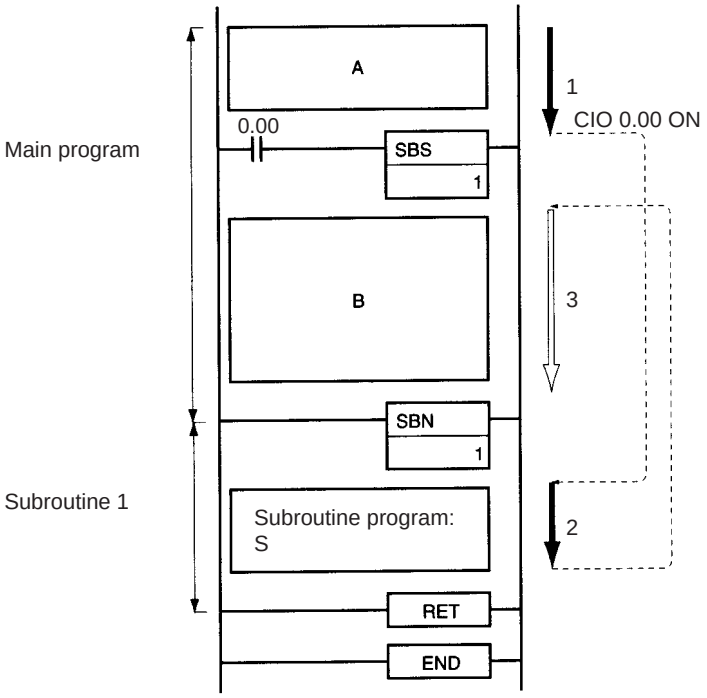
1,2,3...

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

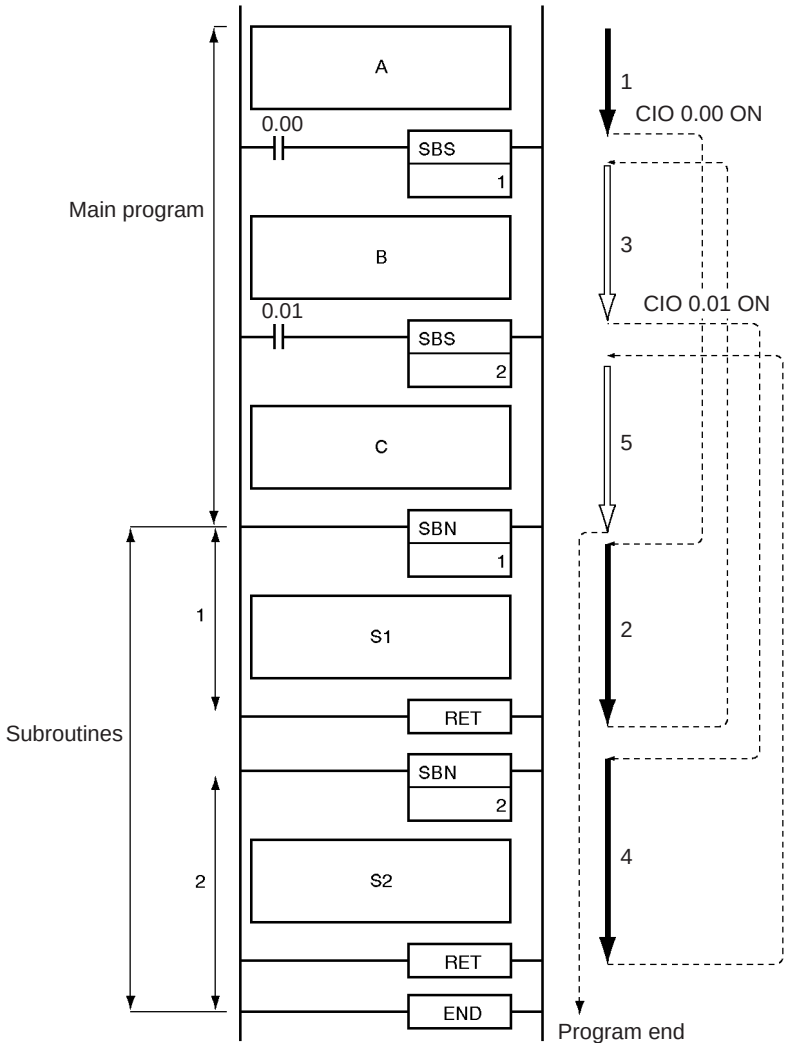
When CIO 0.00 is ON in the following example, subroutine 1 is executed and program execution returns to the next instruction after SBS(091). The remainder of the main program (through the instruction just before SBN(092) 1) is then executed.



0.00	Order of execution
ON	A→S→B
OFF	A→B

Example 2: Sequential (Non-nested) Subroutines

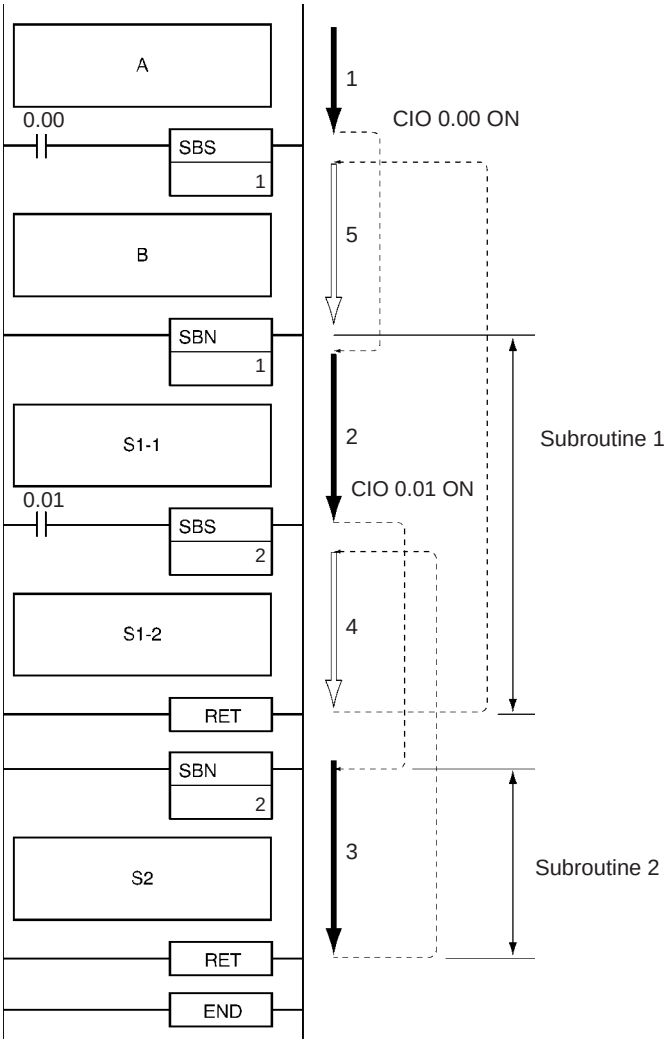
When CIO 0.00 is ON in the following example, subroutine 1 is executed and program execution returns to the next instruction after SBS(091) 1. When CIO 0.01 is ON, subroutine 2 is executed and program execution returns to the next instruction after SBS(091) 2.



0.00	0.01	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 0.00 is ON in the following example, subroutine 1 is executed. If CIO 0.01 is ON, subroutine 2 is executed from within subroutine 1 and program execution returns to the next instruction after SBS(091) 2 when subroutine 2 is completed. Execution of subroutine 1 continues and program execution returns to the next instruction after SBS(091) 1 when subroutine 1 is completed.



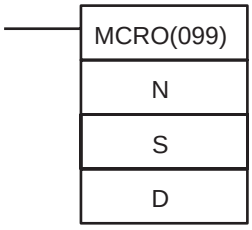
0.00	0.01	Order of execution
ON	ON	A→S1-1→S2→S1-2→B
ON	OFF	A→S1-1→S1-2→B
OFF	ON	A→B
OFF	OFF	A→B

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+3 and the output parameters in D to D+3.

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
Immediate Refreshing Specification		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.

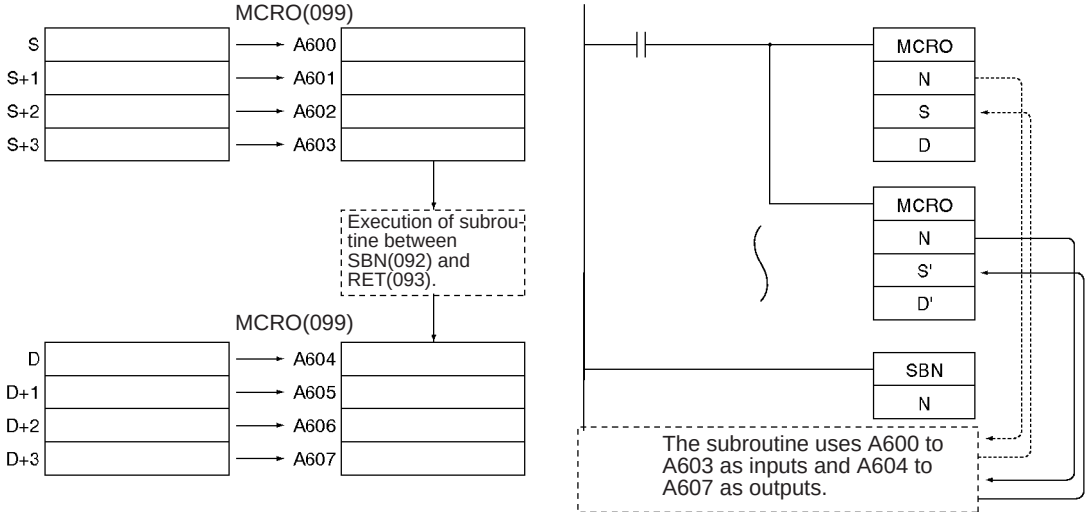
Operand Specifications

Area	N	S	D
CIO Area	---	CIO 0 to CIO 6140	
Work Area	---	W0 to W508	
Holding Bit Area	---	H0 to H508	
Auxiliary Bit Area	---	A0 to A444 A448 to A956	A448 to A956
Timer Area	---	T0000 to T4092	
Counter Area	---	C0000 to C4092	
DM Area	---	D0 to D32764	
Indirect DM addresses in binary	---	@ D0 to @ D32767	
Indirect DM addresses in BCD	---	*D0 to *D32767	
Constants	0 to 255 (decimal)	---	
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 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+3 are copied to A600 through A603 (macro area inputs) and the specified subroutine is executed. When the subroutine is completed, the contents of A604 through A607 (macro area outputs) are copied to D through D+3 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	A600 to A603	When MCRO(099) is executed the four words from S to S+3 are copied to A600 to A603. These input words are passed to the subroutine.
Macro area input words	A604 to A607	After the subroutine specified in MCRO(099) has been executed, the output data in these output words and copied to D to D+3.

Precautions

The four words of input data (words or bits) in A600 to A603 and the four words of output data (words or bits) in A604 to A607 must be used in the subroutine called by MCRO(099). It is not possible to pass more than four words of data.

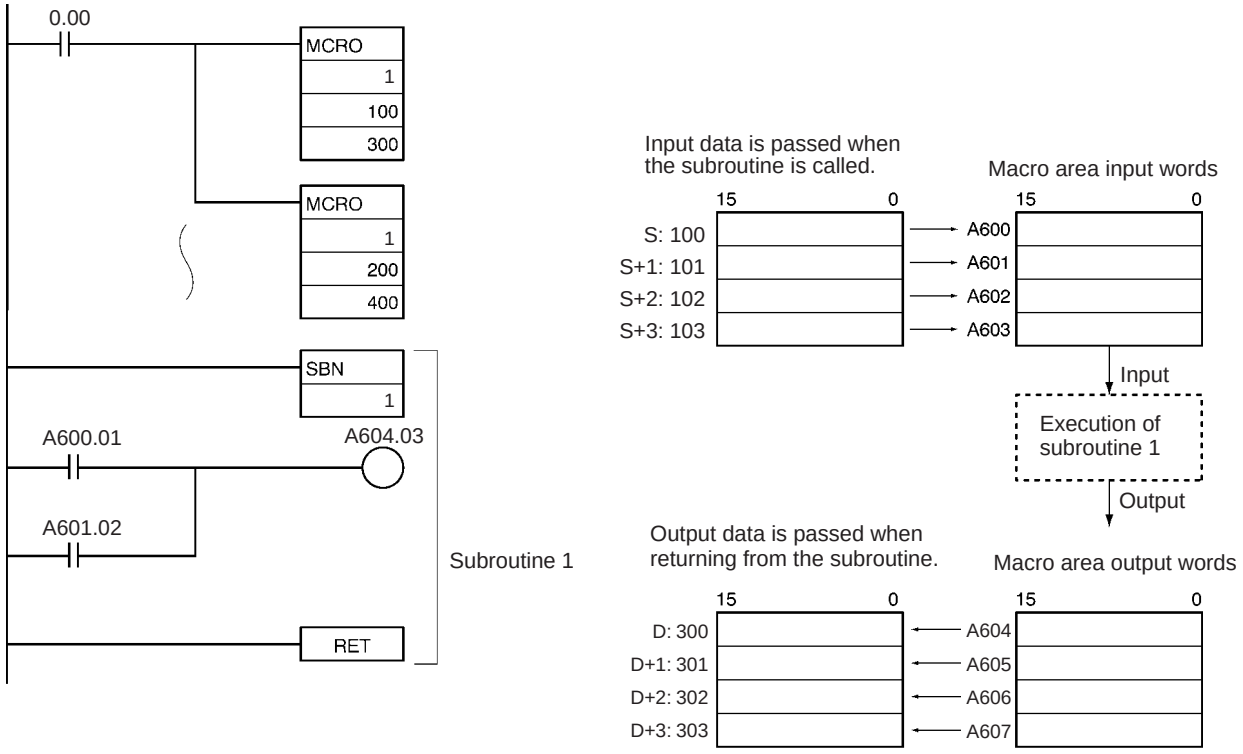
It is possible to nest MCRO(099) instructions, but the data in the macro area input and output words (A600 to A607) must be saved before calling another subroutine because all MCRO(099) instructions use the same 8 words.

Example

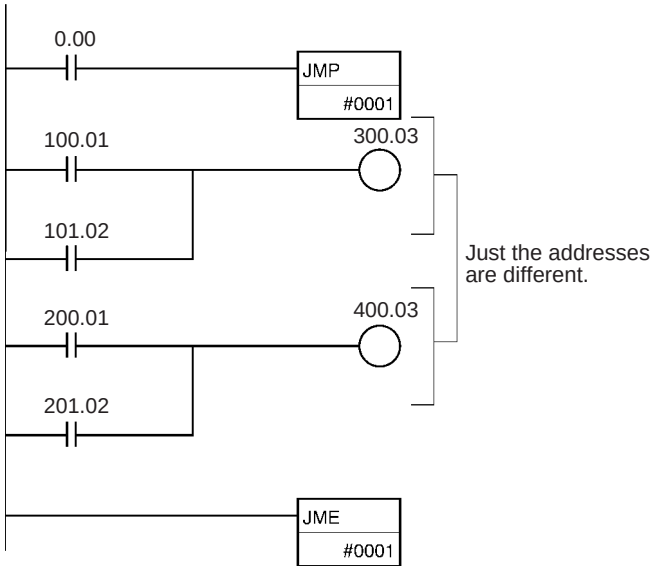
When CIO 0.00 is ON in the following example, two MCRO(099) instructions pass different input and output data to subroutine 1.

1,2,3... 1. The first MCRO(099) instruction passes the input data in CIO 100 to CIO 103 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 300 to CIO 303.

2. The second MCRO(099) instruction passes the input data in CIO 200 to CIO 203 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 400 to CIO 403.



The second MCRO(099) instruction operates in the same way, but the input data in CIO 200 to CIO 203 is passed to A600 to A603 and the output data in A604 to A607 is passed to CIO 400 to CIO 403.

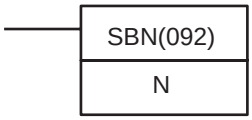


3-18-3 SUBROUTINE ENTRY: SBN(092)

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)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Subroutine number

Specifies the subroutine number between 0 and 255 decimal.

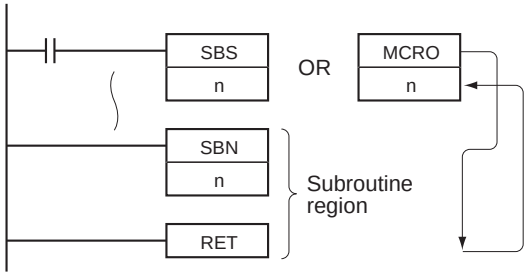
Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit 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 Registers	---
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).

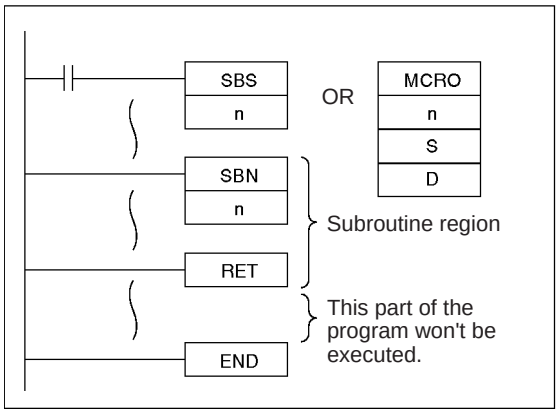
The region of the program beginning at the first SBN(092) instruction is the subroutine region. A subroutine is executed only when it has been called by SBS(091) or MCRO(099).



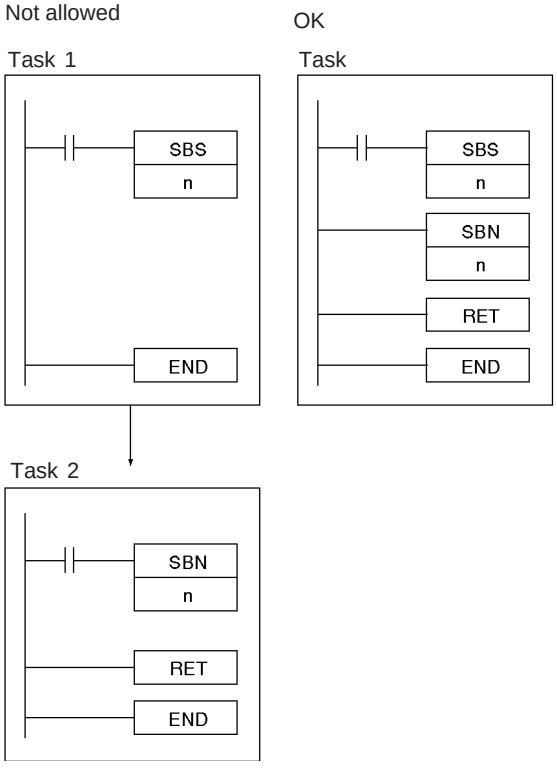
Precautions

When the subroutine is not being executed, the instructions are treated as NOP(000).

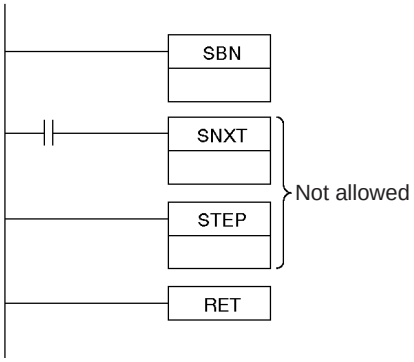
Place the subroutines after the main program and just before the END(001) instruction in the program for each task. If part of the main program is placed after the subroutine region, that program section will be ignored.



Be sure to place each subroutine in the same program (task) as its corresponding SBS(091) 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.

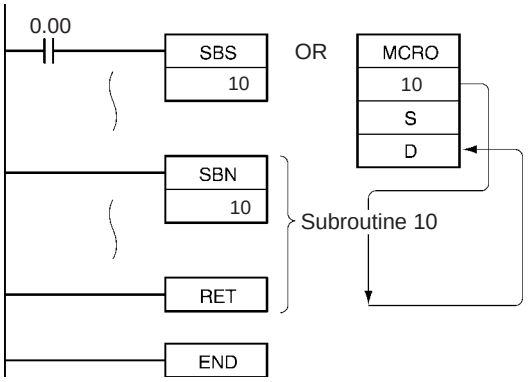


The step instructions, STEP(008) and SNXT(009) cannot be used in subroutines.



Example

When CIO 0.00 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)
Immediate Refreshing Specification		Not supported

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) or MCRO(099) instruction that called the subroutine. When the subroutine has been called by MCRO(099), the output data in A604 through A607 is written to D through D+3 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).

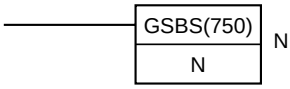
3-18-5 GLOBAL SUBROUTINE CALL: GSBS(750)

Purpose

Calls the global subroutine with the specified subroutine number and executes that program. The same global subroutine can be called from two or more tasks.

GSBS(750) is used in combination with GSBN(751) and GRET(752), the GLOBAL SUBROUTINE ENTRY and GLOBAL SUBROUTINE RETURN instructions.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	GSBS(750)
	Executed Once for Upward Differentiation	@GSBS(750)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Global subroutine number

Specifies the global subroutine number between 0 and 255 decimal.

Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit 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 Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

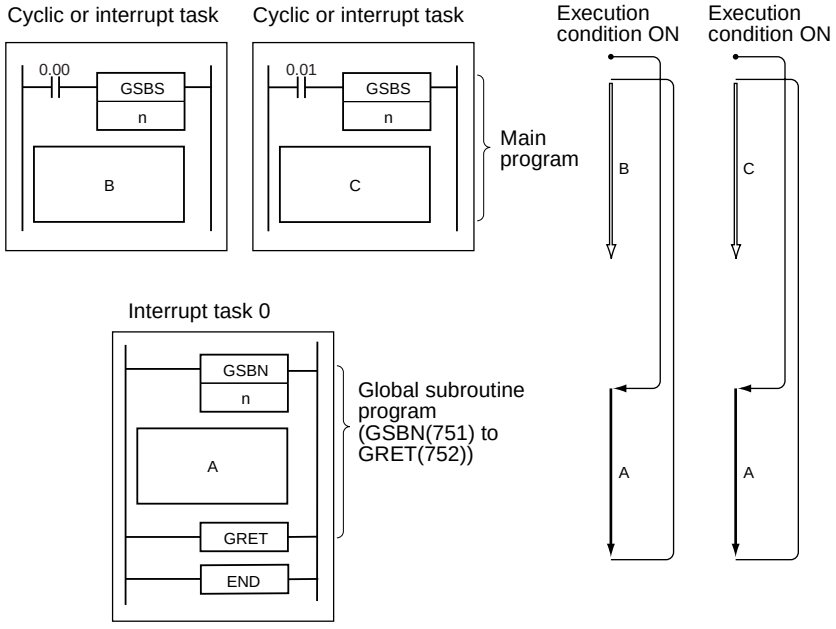
Description

GSBS(750) calls the global subroutine with the specified global subroutine number. The global subroutine is the program section between GSBN(751) and GRET(752). When the global subroutine is completed, program execution continues with the next instruction after GSBS(750).

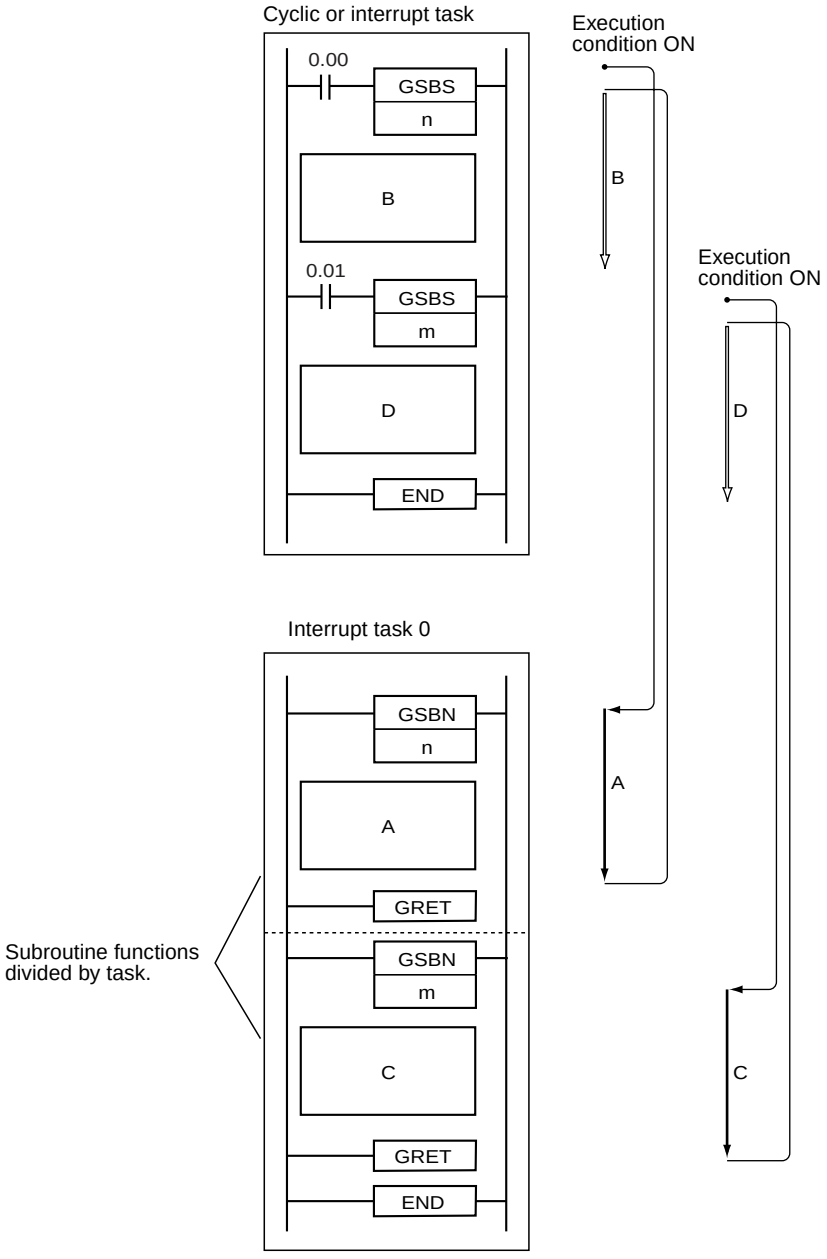
This instruction can be written into multiple tasks with the same global subroutine number to call that program from the different tasks. The program can be modularized by making global subroutines into standard subroutines that are common to many tasks.

The global subroutine region (between GSBN(751) and GRET(752)) must be defined in interrupt task 0. If it is defined in another task, an error will occur and the Error Flag will be turned ON when the GSBS(750) instruction is executed.

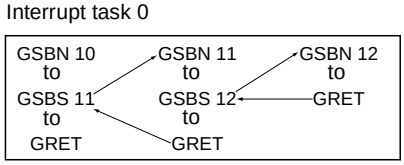
The GSBS(750) instruction can be written in both cyclic tasks (including extra cyclic tasks) and interrupt tasks.



Multiple global subroutine regions (GSBN(751) to GRET(752)) can be defined in interrupt task 0.



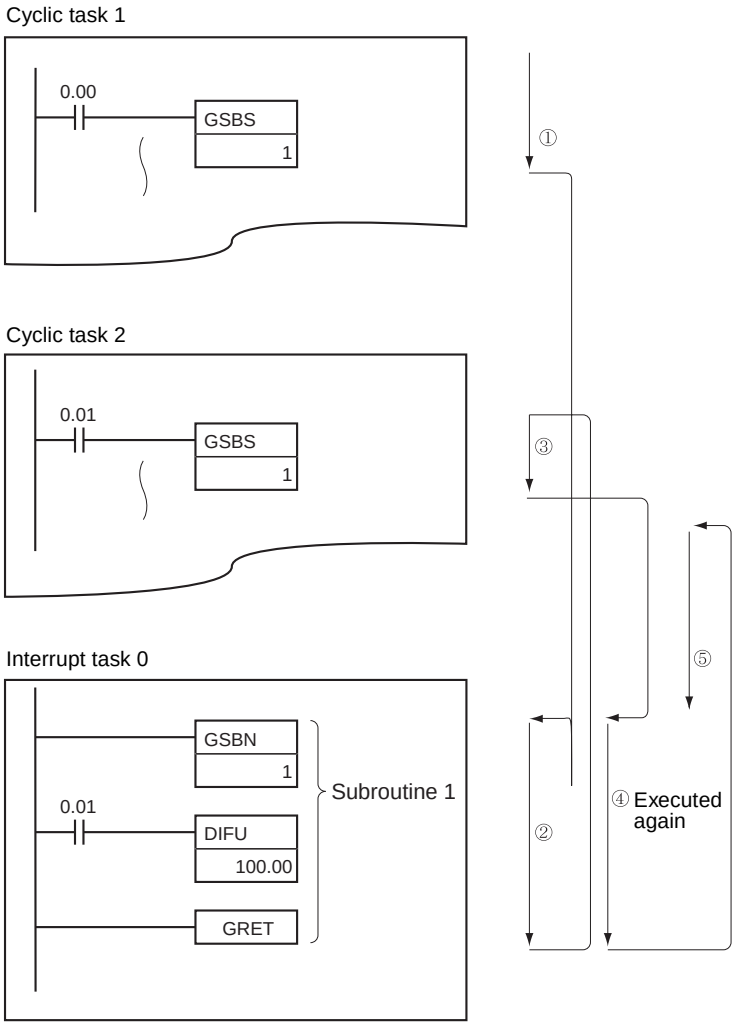
An SBS(091) or GSBS(750) instruction can be written within a subroutine region (SBN(092) to RET(093)) or global subroutine region (GSBN(751) to GRET(752)) to “nest” subroutines. Subroutines can be nested up to 16 levels.



Global Subroutines and Differentiation

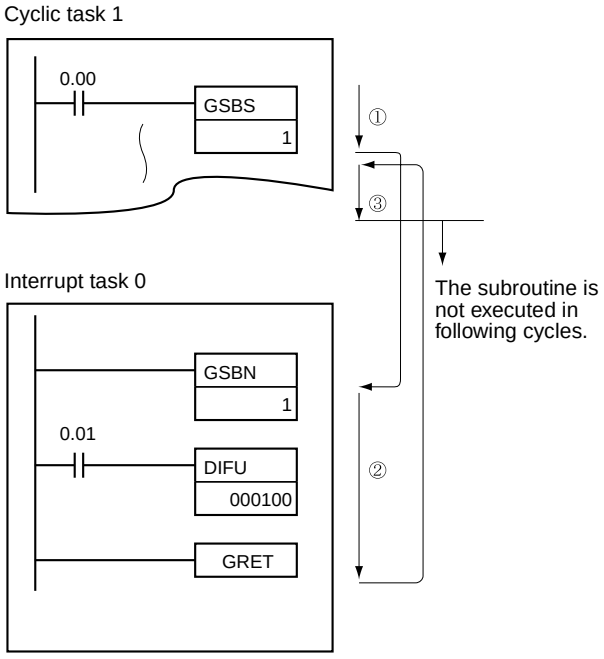
Observe the following precautions when using differentiated instructions (DIFU(013), DIFU(014), or up/down differentiated instructions) in subroutines.

The operation of differentiated instructions in a global subroutine is unpredictable if a subroutine is executed more than once in the same cycle. In the following example, global subroutine 0001 is executed when CIO 0.00 is ON and CIO 100.00 is turned ON by DIFU(013) when CIO 0.01 has gone from OFF to ON. If CIO 0.01 is ON in the same cycle, global subroutine 0001 will be executed again but this time DIFU(013) will not detect the rising edge of CIO 0.01 and CIO 100.00 will be turned OFF.



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 global subroutine was not called a second time.

In the following example, global subroutine 0001 is executed if CIO 0.00 is ON. Output CIO 100.00 is turned ON by DIFU(013) when CIO 0.01 has gone from OFF to ON. If CIO 0.00 is OFF in the following cycle, subroutine 0001 will not be executed again and output CIO 100.00 will remain ON.



Flags

Name	Label	Operation
Error Flag	ER	ON if nesting exceeds 16 levels (counting both regular and global subroutines). ON if the specified global subroutine does not exist. ON if a global subroutine calls itself. ON if a global subroutine being executed is called. ON if the specified subroutine is not defined in interrupt task 0. OFF in all other cases.

Precautions

The GLOBAL SUBROUTINE ENTRY instruction, GSBN(751), and the corresponding GLOBAL SUBROUTINE RETURN instruction, GRET(752) must be programmed in interrupt task 0. If the global subroutine region is not programmed in interrupt task 0, an error will occur and the Error Flag will be turned ON when the GSBS(750) instruction is executed.

The regular SUBROUTINE CALL instruction, SBS(091), cannot call a global subroutine region (GSBN(751) to GRET(752)).

GSBS(750) will not be executed when it is within a program section interlocked by IL(002) and ILC(003), so interlocks are not allowed within global subroutine regions.

The same global subroutine region (GSBN(751) to GRET(752)) can be called more than once.

When GSBS(750) is executed in the following cases, the global subroutine will not actually be called and the Error Flag will be turned ON:

1,2,3...

1. The specified global subroutine is not defined.
2. Subroutine nesting (counting both regular and global subroutines) exceeds 16 levels.
3. The global subroutine is calling itself.
4. The specified global subroutine is being executed.
5. The specified global subroutine is not defined in interrupt task 0.

Examples

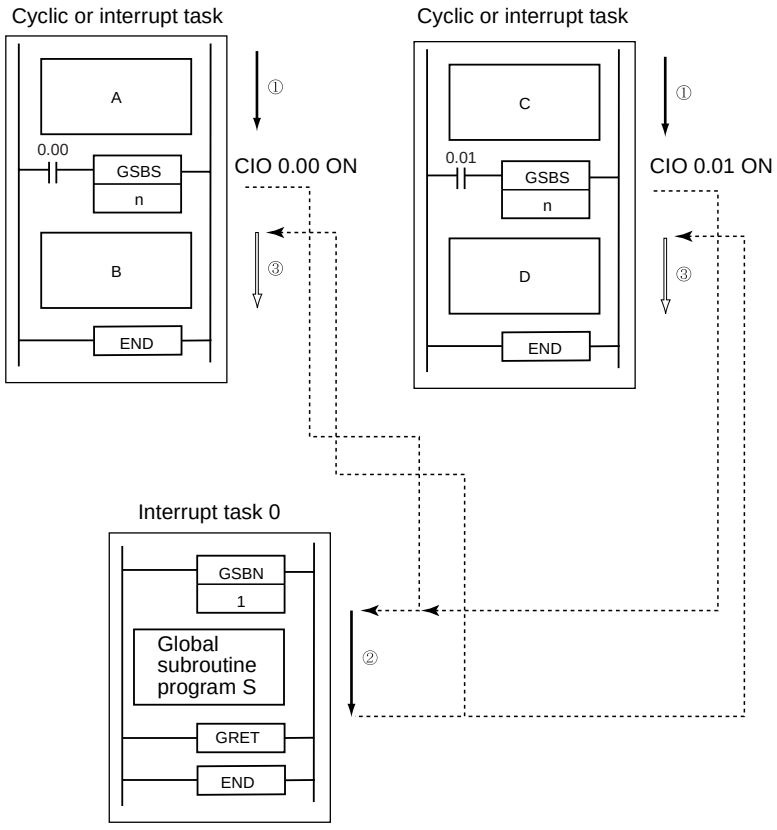
Example 1

When CIO 0.00 is ON in the following example, global subroutine 1 is executed and program execution returns to the next instruction after GSBS(750).

Status of CIO 0.00	Order of program execution
ON	A → S → B
OFF	A → B

When CIO 0.01 is ON in the following example, global subroutine 1 is executed and program execution returns to the next instruction after GSBS(750).

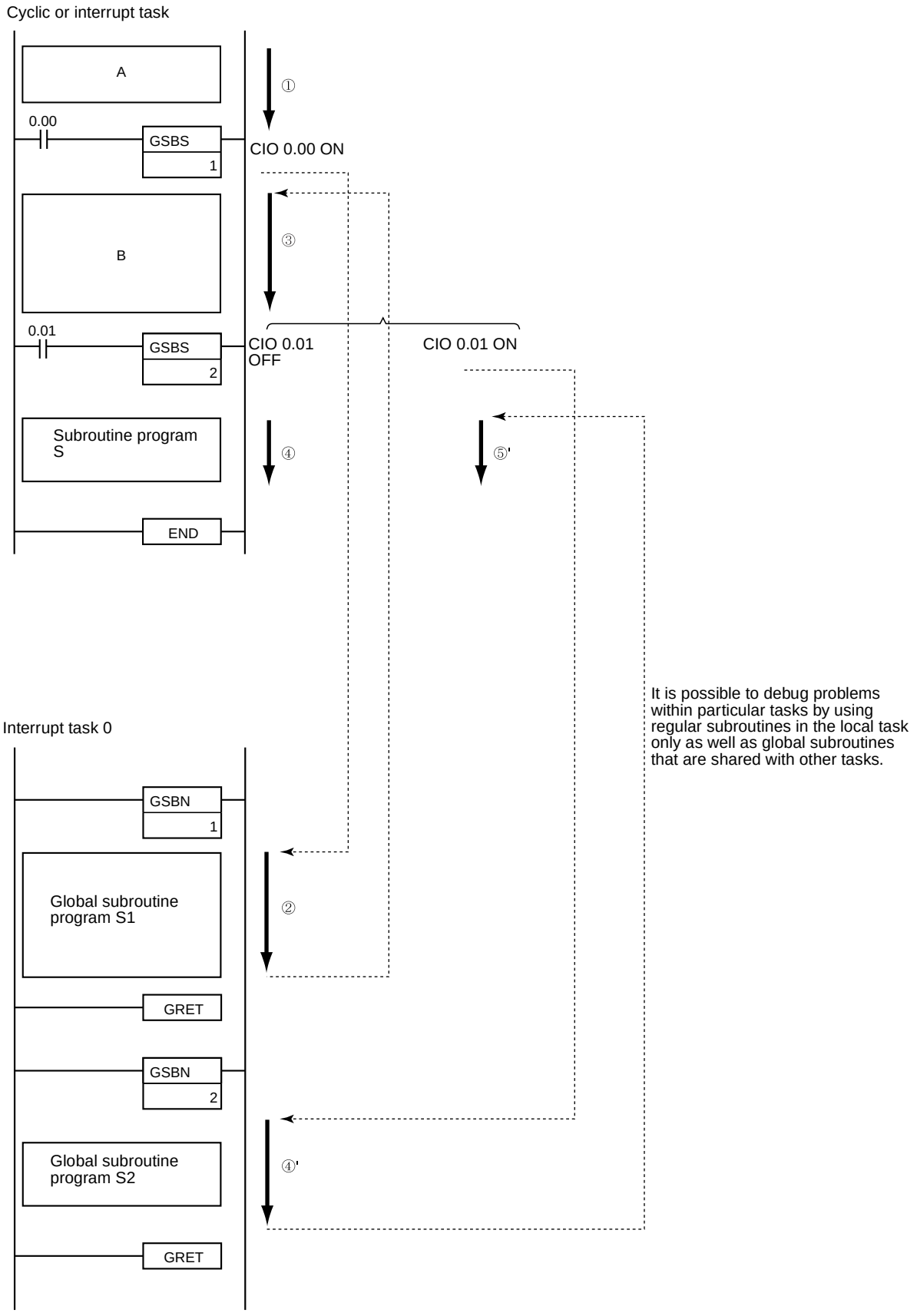
Status of CIO 0.01	Order of program execution
ON	C → S → D
OFF	C → D



Example 2

Two or more global subroutine programs can be programmed in interrupt task 0. In this case, interrupt task 0 can be divided and used as the subroutine function's task.

When CIO 0.00 is ON, global subroutine program 1 is executed.
When CIO 0.01 is ON, global subroutine program 2 is executed.



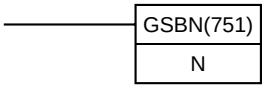
3-18-6 GLOBAL SUBROUTINE ENTRY: GSBN(751)

Purpose

Indicates the beginning of the global subroutine program with the specified subroutine number. Used in combination with GRET(752) to define a global subroutine region.

GSBN(751) is used in combination with GSBS(750) and GRET(752), the GLOBAL SUBROUTINE CALL and GLOBAL SUBROUTINE RETURN instructions.

Ladder Symbol



Variations

Variations	Executed Each Cycle for ON Condition	GSBN(751)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Global subroutine number

Specifies the global subroutine number between 0 and 255 decimal.

Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit 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 Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

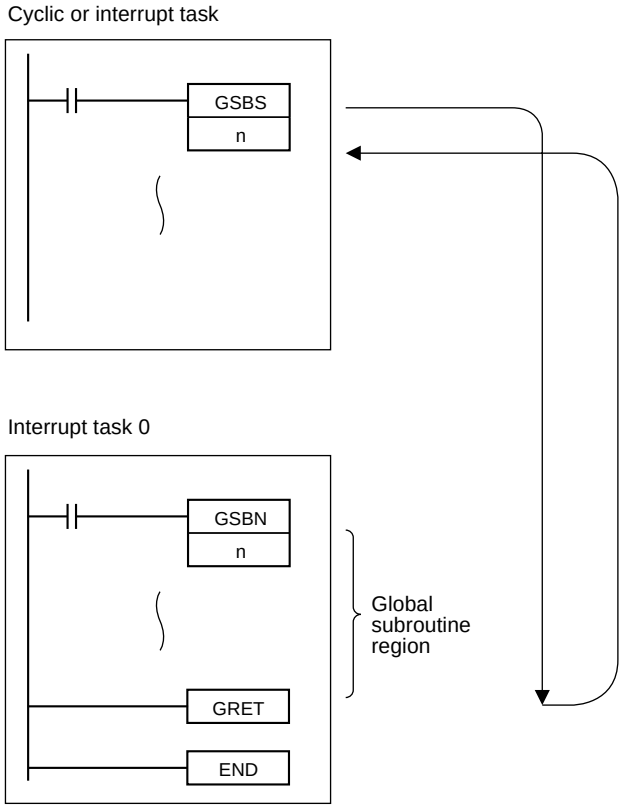
Description

GSBN(751) indicates the beginning of the global subroutine with the specified subroutine number. The end of the subroutine is indicated by GRET(752).

The region of the program beginning at the first GSBN(751) instruction is the subroutine region. A subroutine is executed only when it has been called by GSBS(750).

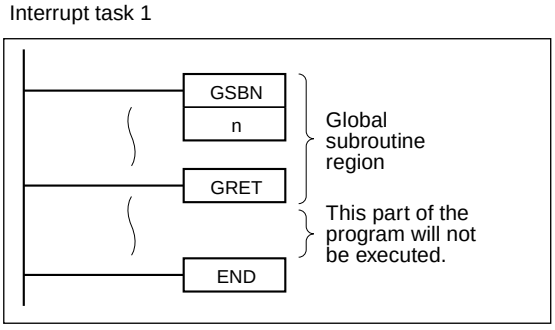
The global subroutine region (between GSBN(751) and GRET(752)) must be defined in interrupt task 0. If it is defined in another task, an error will occur and the Error Flag will be turned ON when the GSBS(750) instruction is executed.

The GSBS(750) instruction can be written both cyclic tasks (including extra cyclic tasks) and interrupt tasks.



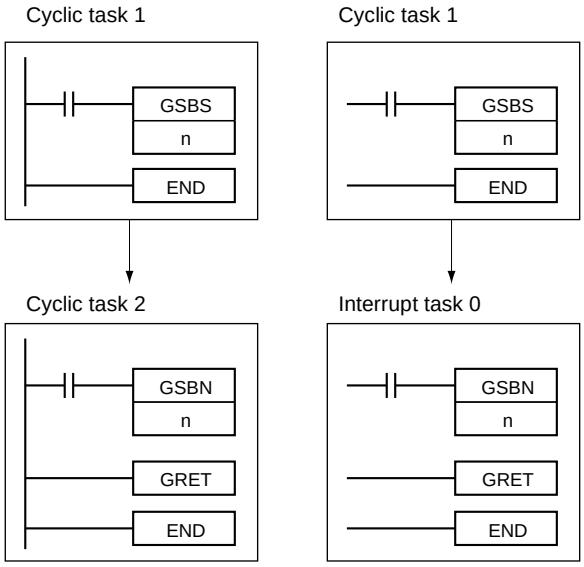
Precautions

- When the subroutine is not being executed, the instructions are treated as NOP(000).
- Place the global subroutine region (GSBN(751) to GRET(752)) in interrupt task 0 just before the END(001) instruction. When two or more global subroutines are being used, group them together in interrupt task 0 after the end of the main program. If part of the main program is placed after the global subroutine region, that program section will be ignored.

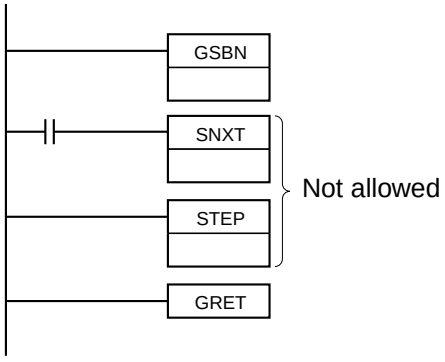


- Always place the global subroutines in interrupt task 0. An error will occur if a global subroutine is called and the subroutine is not in interrupt task 0.

Not allowed OK

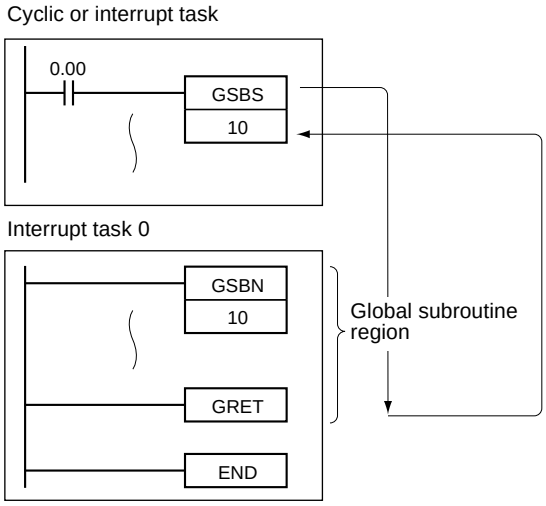


- The step instructions, STEP(008) and SNXT(009) cannot be used in global subroutines.



Example

When CIO 0.00 is ON in the following example, global subroutine 10 is executed and program execution returns to the next instruction after the GSBS(750) instruction that called the subroutine.



3-18-7 GLOBAL SUBROUTINE RETURN: GRET(752)

Purpose

Indicates the end of a subroutine program. Used in combination with GSBN(751) to define a subroutine region.

GRET(752) is used in combination with GSBS(750) and GSBN(751), the GLOBAL SUBROUTINE CALL and GLOBAL SUBROUTINE ENTRY instructions.

Ladder Symbol

Variations

Variations	Executed Each Cycle for ON Condition	GRET(752)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

GRET(752) indicates the end of a global subroutine and GSBN(751) indicates the beginning of a global subroutine. See *3-18-6 GLOBAL SUBROUTINE ENTRY: GSBN(751)* for more details on the operation of global subroutines.

When program execution reaches GRET(752) it is automatically returned to the next instruction after the GSBS(750) instruction that called the global subroutine.

Precautions

When the subroutine is not being executed, the instructions are treated as NOP(000).

Example

See *3-18-6 GLOBAL SUBROUTINE ENTRY: GSBN(751)* for examples of the operation of GRET(752).

3-19 Interrupt Control Instructions

This section describes instructions used to control interrupts.

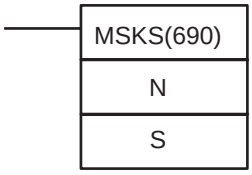
Instruction	Mnemonic	Function code	Page
SET INTERRUPT MASK	MSKS	690	693
READ INTERRUPT MASK	MSKR	692	697
CLEAR INTERRUPT	CLI	691	700
DISABLE INTERRUPTS	DI	693	703
ENABLE INTERRUPTS	EI	694	704

3-19-1 SET INTERRUPT MASK: MSKS(690)

Purpose

Both input interrupt tasks and scheduled interrupt tasks are masked (disabled) and the internal timer for scheduled interrupts is stopped when the PLC enters RUN mode. MSKS(690) can be used to unmask or mask input interrupts and set the time intervals for scheduled interrupts.

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

Input Interrupts

N specifies the input interrupt number and the function of MSKS(690) and S specifies operational details.

Operand		Contents	
		Specify ON or OFF to generate interrupt	Mask or unmask interrupt
N	Input interrupt 0 (interrupt task 140)	110 (or 10)	100 (or 6)
	Input interrupt 1 (interrupt task 141)	111 (or 11)	101 (or 7)
	Input interrupt 2 (interrupt task 142)	112 (or 12)	102 (or 8)
	Input interrupt 3 (interrupt task 143)	113 (or 13)	103 (or 9)
	Input interrupt 4 (interrupt task 144)	114	104
	Input interrupt 5 (interrupt task 145)	115	105
	Input interrupt 6 (interrupt task 146)	116	106
	Input interrupt 7 (interrupt task 147)	117	107
S		0000 hex: Detect ON (default) 0001 hex: Detect OFF	0000 hex: Unmask, direct mode 0001 hex: Mask 0002 hex: Unmask, counter mode, start decrementing 0003 hex: Unmask, counter mode, start incrementing

Note Input interrupts 6 and 7 cannot be used on Y CPU Units.

Scheduled Interrupts

N specifies the scheduled interrupt number and the starting method and S specifies the interrupt interval.

Operand			Contents
N	Scheduled interrupt 0 (interrupt task 2)		14 hex: Reset start (Reset internal timer and start timing.) 4 hex: Start without reset (Specify the time to the first interrupt separately with CLI(691).)
S			0000 hex: Prohibit schedule interrupts and stop internal timer.
	PLC Setup parameter (Scheduled Interrupt Interval)	10 ms	Scheduled interrupt interval: 0001 to 270F hex (10 to 99,990 ms)
		1 ms	Scheduled interrupt interval: 0001 to 270F hex (1 to 9,999 ms)
		0.1 ms	Scheduled interrupt interval: 0005 to 270F hex (5 to 999.9 ms) **Note** An error will occur if 0001 to 0004 hex is set.

Operand Specifications

Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511

Area	N	S
Auxiliary Bit Area	---	A0 to A447 A448 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ 32767
Indirect DM addresses in BCD	---	*D0 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

MSKS(690) controls input interrupts and scheduled interrupts. The value of N identifies the interrupt.

Input Interrupts: N = 100 to 107, 110 to 117, or 6 to 13

- MSKS(690) specifies whether interrupts are generated when the interrupt input turns ON or turns OFF and whether to mask or unmask the interrupt. If the specification is omitted, interrupts are generated when the interrupt input turns ON.
- When an interrupt is unmasked, either direct mode or counter mode (incrementing or decrementing) is specified. Refer to *5-1 Interrupt Functions* in the *CP1H Operation Manual* for details.
- Any interrupts that are masked will be cleared when the interrupt is unmasked or the ON/OFF specification for generating interrupts is changed.

Scheduled Interrupt: N = 4 or 14

- MSKS(690) specifies the interrupt interval and starts the internal timer. The interrupt interval also depends on the setting of the *Scheduled Interrupt Interval* in the PLC Setup.
- The internal timer can be reset or not reset depending on the operands for MSKS(690).
- When the internal timer is reset, timing will start after the timer is reset and scheduled interrupt will occur at the interval specified in S from the time MSKS(690) is executed.
- When the internal timer is not reset, the internal timer will continue operating from the present time and the time to the first interrupt is specified separately with CLI(691). If the time to the first interrupt is not specified with CLI(691), the time to the first interrupt will be undefined, but scheduled interrupts will be started at the latest after two scheduled interrupt intervals have elapsed.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range. For input interrupts: ON if S is not within the specified range of 0000 to 0003. For scheduled interrupts: ON if S is not within the specified range of 0000 to 270F hex when the scheduled interrupt interval is in 10 or 1 ms units or 0005 to 270F hex when the interval is in 0.1 ms units. OFF in all other cases.
Equals Flag	=	OFF
Negative Flag	N	OFF

The following table shows relevant flags in the Auxiliary Area.

Name	Address	Operation
Interrupt Task Error Flag	A402.13	ON in the following case: IORF(097) was executed in an interrupt task without disabling Special I/O cyclic refreshing.
Interrupt Task Error Cause Flag	A426.15	Indicates whether Interrupt Task Error 1 or 2 occurred.
Interrupt Task Error Task Number	A426.00 to A426.11	Indicates the unit number of the Special I/O Unit where the multiple I/O refreshing occurred.

Precautions

Be sure that the time interval is longer than the time required to execute the scheduled interrupt task.

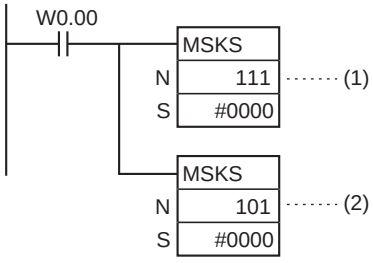
When IORF(097) is being executed within an interrupt task to refresh I/O in a Special I/O Unit, cyclic refreshing with that Special I/O Unit must be disabled in the PLC Setup. If cyclic refreshing with the Special I/O Unit is not disabled, IORF(097) might be executed during cyclic refreshing resulting in a non-fatal Duplicate Refresh Error and turning ON the Interrupt Task Error Flag (A402.13).

A440 contains the maximum processing time for interrupt tasks and the rightmost byte of A441 contains the interrupt task number of the task with the longest processing time.

Examples

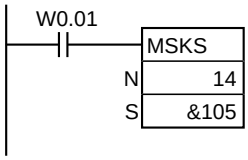
Enabling Input Interrupts in Direct Mode

When W0.00 turns ON in the following example, the first MSKS(690) (1) specifies generating input interrupts for input interrupt 1 (CIO 0.01) when the interrupt input turns ON and the second MSKS(690) (2) unmasks the interrupt.



Starting Scheduled Interrupts while Resetting the Scheduled Interrupt Timer

When W0.01 turns ON in the following example, MSKS(690) sets the schedule interrupt interval for schedule interrupt 0 to 10.5 ms (assuming the unit is set to 0.1 ms in the PLC Setup), resets the internal timer, and starts the internal timer.

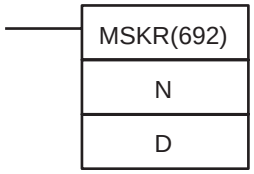


3-19-2 READ INTERRUPT MASK: MSKR(692)

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

Input Interrupts

N specifies the input interrupt number and the data to read and D specifies the storage location for the read data.

Operand		Contents	
		Read ON or OFF to generate interrupt	**Read mask status**
N	Input interrupt 0 (interrupt task 140)	110 (or 10)	100 (or 6)
	Input interrupt 1 (interrupt task 141)	111 (or 11)	101 (or 7)
	Input interrupt 2 (interrupt task 142)	112 (or 12)	102 (or 8)
	Input interrupt 3 (interrupt task 143)	113 (or 13)	103 (or 9)
	Input interrupt 4 (interrupt task 144)	114	104
	Input interrupt 5 (interrupt task 145)	115	105
	Input interrupt 6 (interrupt task 146)	116	106
	Input interrupt 7 (interrupt task 147)	117	107
D		0000 hex: Detect ON (default) 0001 hex: Detect OFF	0000 hex: Unmask, direct mode 0001 hex: Mask 0002 hex: Unmask, counter mode, start decrementing 0003 hex: Unmask, counter mode, start incrementing

Note Input interrupts 6 and 7 cannot be used on Y CPU Units.

Scheduled Interrupts

N specifies the scheduled interrupt number and the data to read and D specifies the storage location for the read data.

Operand			Contents	
			Read scheduled interrupt interval	**Read present value of internal timer (i.e., time from first interrupt or previous interrupt processing)**
N			4	7
D			0000 hex: Scheduled interrupts prohibited.	0000 to 270F hex (1 to 9999): Present value of internal timer
			0001 to 270F hex (1 to 9999): Scheduled interrupt interval	
	PLC Setup parameter (Scheduled Interrupt Interval)	10 ms	10 to 99.990 ms	0 to 99.990 ms
		1 ms	1 to 9.999 ms	0 to 9.999 ms
		0.1 ms	0.1 to 999.9 ms	0 to 999.9 ms

Note If the scheduled interrupt has been prohibited, he time elapsed until the schedule interrupt's internal timer was stopped can be read. The present value will be 0 if the scheduled interrupt has never been started.

Operand Specifications

Area	N	D
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A448 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 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

The value of N identifies the interrupt.

Input Interrupts: N = 100 to 107, 110 to 117, or 6 to 13

The mask status or the trigger specification (ON or OFF) specified with N is stored in D.

Scheduled Interrupt: N = 4 or 14

The scheduled interrupt interval (set value) or the present value of the internal timer specified with N is stored in D as a hexadecimal value. The units for the scheduled interrupt interval is specified in the PLC Setup as the *Scheduled Interrupt Interval.*

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 5 (0 to 15 for the CJ1M). OFF in all other cases.

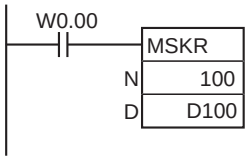
Precautions

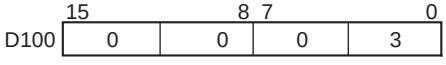
MSKR(692) can be executed in the main program or in interrupt tasks.

Examples

Input Interrupts

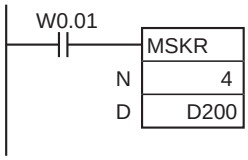
When W0.00 turns ON in the following example, the mask status of input interrupt 0 (CIO 0.00) is stored in D100. The value in the example (0003) says that the interrupt is unmasked in incrementing counter mode.

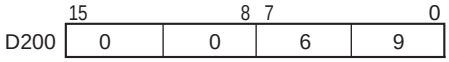




Scheduled Interrupts

When W0.01 turns ON in the following example, the scheduled interrupt interval is stored in D200. The value in the example (0069) says that the interval is 10.5 ms (0069 hex = 105 decimal) assuming that the schedule interrupt interval unit is set to 0.1 ms in the PLC Setup.



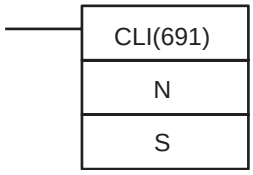


3-19-3 CLEAR INTERRUPT: CLI(691)

Purpose

Clears or retains recorded interrupt inputs for input interrupts and high-speed counter interrupts, or sets the time to the first scheduled interrupt for scheduled interrupts.

Ladder Symbol



N: Interrupt identifier

S: Interrupt 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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

Input Interrupts

N specifies the input interrupt number and S specifies the operation.

Operand	Contents
N	Input Interrupt Number 100 or 6: Input interrupt 0 (interrupt task 140) 101 or 7: Input interrupt 1 (interrupt task 141) 102 or 8: Input interrupt 2 (interrupt task 142) 103 or 9: Input interrupt 3 (interrupt task 143) 104: Input interrupt 4 (interrupt task 144) 105: Input interrupt 5 (interrupt task 145) 106: Input interrupt 6 (interrupt task 146) 107: Input interrupt 7 (interrupt task 147)
S	Recorded Interrupt Clear Specification 0000 hex: Recorded interrupt retained 0001 hex: Recorded interrupt cleared

Note Input interrupts 6 and 7 cannot be used on Y CPU Units.

Scheduled Interrupts

N specifies the scheduled interrupt number and S specifies the time to the first scheduled interrupt.

Operand	Contents
N	Specify the scheduled interrupt number. 4: Scheduled interrupt 0 (interrupt task 2)
S	0000 to 270F hex: Time to first scheduled interrupt (0 to 9999) **Note** The unit for the scheduled interrupt interval can be set to 10 ms, 1.0 ms, or 0.1 ms in the PLC Setup interrupt settings.

High-speed Counter Interrupts

N specifies the high-speed counter interrupt number and S specifies the operation

Operand	Contents
N	High-speed Counter Interrupt Number 10: High-speed counter input 0 11: High-speed counter input 1 12: High-speed counter input 2 13: High-speed counter input 3
S	Recorded Interrupt Clear Specification 0000 hex: Recorded interrupt retained 0001 hex: Recorded interrupt cleared

Operand Specifications

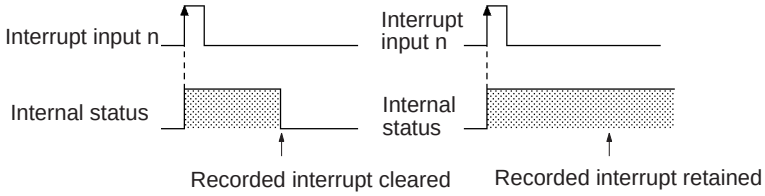
Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	DR0 to DR15
Data Registers	Specified values only	
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

Depending on the value of N, CLI(691) either clears the specified recorded input interrupts or high-speed counter interrupts, or sets the time before execution of the first scheduled interrupt.

Input Interrupts: N = 100 to 107 or 6 to 9

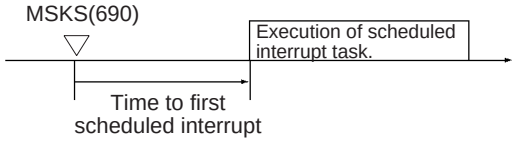
Recorded interrupts are either cleared or retained according to the value of S for the input interrupt specified by N.



If an input interrupt task is being executed and an interrupt input with a different interrupt number is received, that interrupt number is recorded internally. The recorded input 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.

Scheduled Interrupts: N = 4

N is 4, the content of S specifies the time interval to the first scheduled interrupt task after MSKS(690) is executed.



High-speed Counter Interrupts: N = 10 to 13

Recorded interrupts are either cleared or retained according to the value of S for the high-speed counter interrupt specified by N.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range. ON if S is not 0000 or 0001 hex (for high-speed counter interrupts and input interrupts only). ON if S is not within the specified range of 0000 to 270F hex for scheduled interrupts. OFF in all other cases.

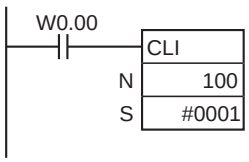
Precautions

A new interrupt input will be ignored if that interrupt has already been recorded. Furthermore, a new interrupt input will be ignored if it is received while its interrupt task is being executed.

Examples

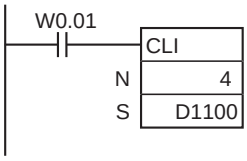
Input Interrupts

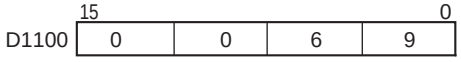
When W0.00 turns ON in the following example, CLI(691) clears all interrupts stored for input interrupt 0.



Scheduled Interrupts

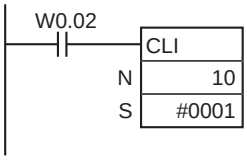
When W0.01 turns ON in the following example, CLI(691) sets the time to the first schedule interrupt 10.5 ms (0069 hex = 105 decimal) assuming that the schedule interrupt interval unit is set to 0.1 ms in the PLC Setup.





High-speed Counter Interrupts

When W0.02 turns ON in the following example, CLI(691) clears all interrupts stored for high-speed counter interrupt 0.

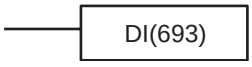


3-19-4 DISABLE INTERRUPTS: DI(693)

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
Immediate Refreshing Specification		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 interrupts, scheduled interrupts, high-speed counter interrupts, and external interrupts).

All interrupt tasks will be disabled until they are enabled again by execution of EI(694).

Flags

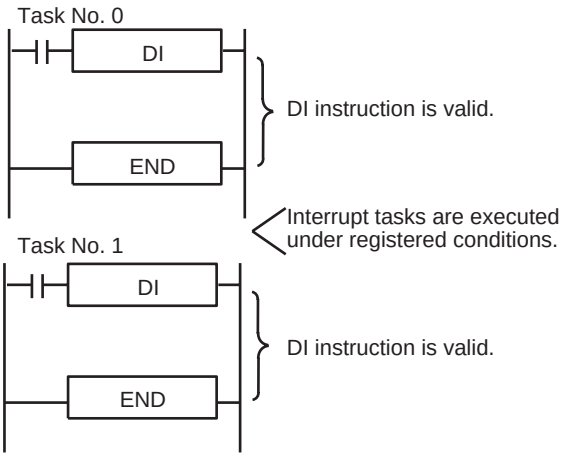
Name	Label	Operation
Error Flag	ER	ON if DI(693) is executed from an interrupt task. OFF in all other cases.

Precautions

All interrupt tasks will remain disabled until EI(694) is executed.

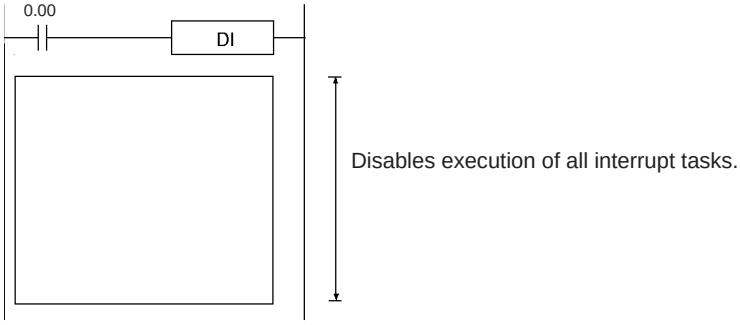
DI(693) cannot be executed from an interrupt task.

DI(693) cannot be executed for more than one cyclic task at a time. To disable more than one cycle execution task, insert DI(693) in each cyclic task. Any interrupts that occur while one cycle execution task is being executed will be executed after the cycle execution task has been completed as shown in the following example unless they are disabled by CLI(691).



Examples

When CIO 0.,00 is ON in the following example, DI(693) disables all interrupt tasks.

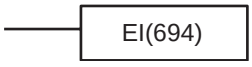


3-19-5 ENABLE INTERRUPTS: EI(694)

Purpose

Enables execution of all interrupt tasks that were disabled with DI(693).

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
Immediate Refreshing Specification		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 temporarily enable all interrupt tasks that were disabled by DI(693). DI(693) disables all interrupts (input interrupts, scheduled interrupts, high-speed counter interrupts, and external interrupts).

Flags

Name	Label	Operation
Error Flag	ER	ON if EI(694) is executed from an interrupt task. OFF in all other cases.

Precautions

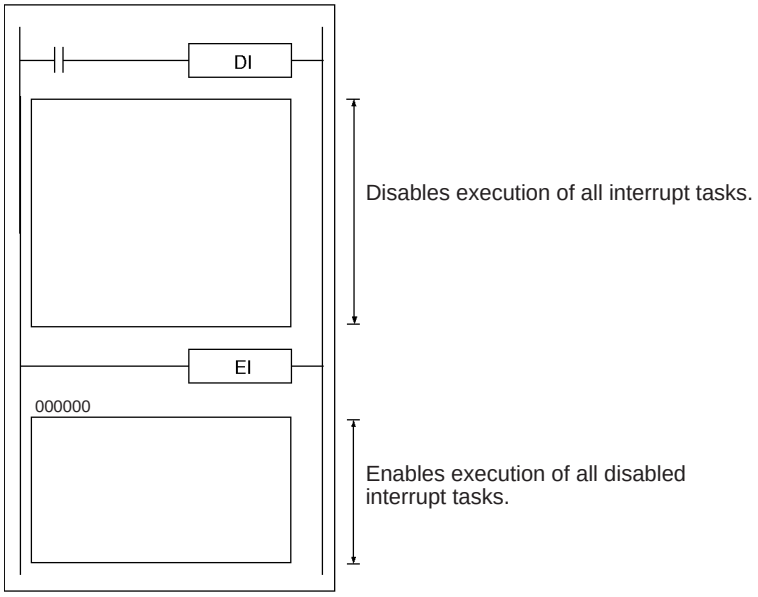
EI(694) does not require an execution condition. It is always executed with an ON execution condition.

EI(694) cannot unmask input interrupts that have not been unmasked by MSKS(690) or set scheduled interrupts that have not been set by MSKS(690).

EI(694) cannot be executed in an interrupt task.

Examples

In the following example, EI(694) enables all interrupt tasks that were disabled by DI(693).



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	706
HIGH-SPEED COUNTER PV READ	PRV	881	710
COUNTER FREQUENCY CONVERT	PRV2	881	716
REGISTER COMPARISON TABLE	CTBL	882	720
SPEED OUTPUT	SPED	885	724
SET PULSES	PULS	886	729
PULSE OUTPUT	PLS2	887	731
ACCELERATION CONTROL	ACC	888	739
ORIGIN SEARCH	ORG	889	745
PULSE WITH VARIABLE DUTY FACTOR	PWM	891	749

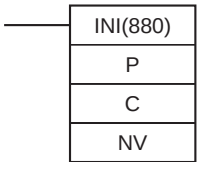
3-20-1 MODE CONTROL: INI(880)

Purpose

INI(880) can be used to execute the following operations for built-in I/O:

- To start comparison with the high-speed counter comparison table
- To stop comparison with the high-speed counter comparison table
- To change the PV of the high-speed counter.
- To change the PV of interrupt inputs in counter mode.
- To change the PV of the pulse output (origin fixed at 0).
- To stop pulse output.

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
Immediate Refreshing Specification		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
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3
0010 hex	High-speed counter 0

P	Port
0011 hex	High-speed counter 1
0012 hex	High-speed counter 2
0013 hex	High-speed counter 3
0100 hex	Interrupt input 0 in counter mode
0101 hex	Interrupt input 1 in counter mode
0102 hex	Interrupt input 2 in counter mode
0103 hex	Interrupt input 3 in counter mode
0104 hex	Interrupt input 4 in counter mode
0105 hex	Interrupt input 5 in counter mode
0106 hex	Interrupt input 6 in counter mode
0107 hex	Interrupt input 7 in counter mode
1000 hex	PWM output 0
1001 hex	PWM output 1

Note Input interrupts 6 and 7 cannot be used on Y CPU Units.

C: Control Data

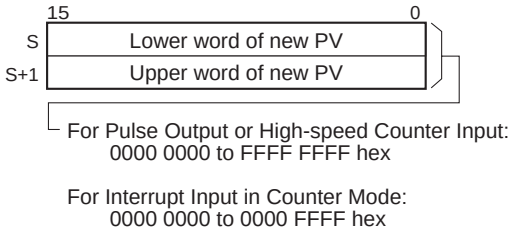
The function of INI(880) is determined by the control data, C.

C	INI(880) function
0000 hex	Starts comparison.
0001 hex	Stops comparison.
0002 hex	Changes the PV.
0003 hex	Stops pulse output.

NV: First Word with New PV

NV and NV+1 contain the new PV when changing the PV.

If C is 0002 hex (i.e., when changing a PV), NV and NV+1 contain the new PV. Any values in NV and NV+1 are ignored when C is not 0002 hex.



Operand Specifications

Area	P	C	NV
CIO Area	---	---	CIO 0 to CIO 6142
Work Area	---	---	W0 to W510
Holding Bit Area	---	---	H0 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D0 to D32766
Indirect DM addresses in binary	---	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	---	*D0 to *D32767

Area	P	C	NV
Constants	See description of operand.	See description of operand.	---
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 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
0000 to 0003 hex: Pulse output	Not allowed.	Not allowed.	OK	OK
0010 to 0013 hex: High-speed counter input	OK	OK	OK	Not allowed.
0100 to 0107 hex: Interrupt input in counter mode	Not allowed.	Not allowed.	OK	Not allowed.
1000 or 1001 hex: PWM output	Not allowed.	Not allowed.	Not allowed.	OK

■ **Starting Comparison (C = 0000 hex)**

If C is 0000 hex, INI(880) starts comparison of a high-speed counter's PV to the comparison table registered with CTBL(882).

Note A target value comparison table must be registered in advance with CTBL(882). If INI(880) is executed without registering a table, the Error Flag will turn ON.

■ **Stopping Comparison (C = 0001 hex)**

If C is 0001 hex, INI(880) stops comparison of a high-speed counter's PV to the comparison table registered with CTBL(882).

■ **Changing a PV (C = 0002 hex)**

If C is 0002 hex, INI(880) changes a PV as shown in the following table.

Port and mode			Operation	Setting range
Pulse output (P = 0000 to 0003 hex)			The present value of the pulse output is changed. The new value is specified in NV and NV+1. **Note** This instruction can be executed only when pulse output is stopped. An error will occur if it is executed during pulse output.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
High-speed counter input (P = 0010 to 0013 hex)	Linear Mode	Differential inputs, increment/ decrement pulses, or pulse + direction inputs	The present value of the high-speed counter is changed. The new value is specified in NV and NV+1. **Note** An error will occur for the instruction if the specified port is not set for a high-speed counter.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
		Increment pulse input		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
	Ring Mode			0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
Interrupt inputs in counter mode (P = 0100 to 0107 hex)			The present value of the interrupt input is changed. The new value is specified in NV and NV+1.	0000 0000 to 0000 FFFF hex (0 to 65,535) **Note** An error will occur if a value outside this range is specified.

■ **Stopping Pulse Output (P = 0000 to 0003, 1000, or 1001 hex and C = 0003 hex)**

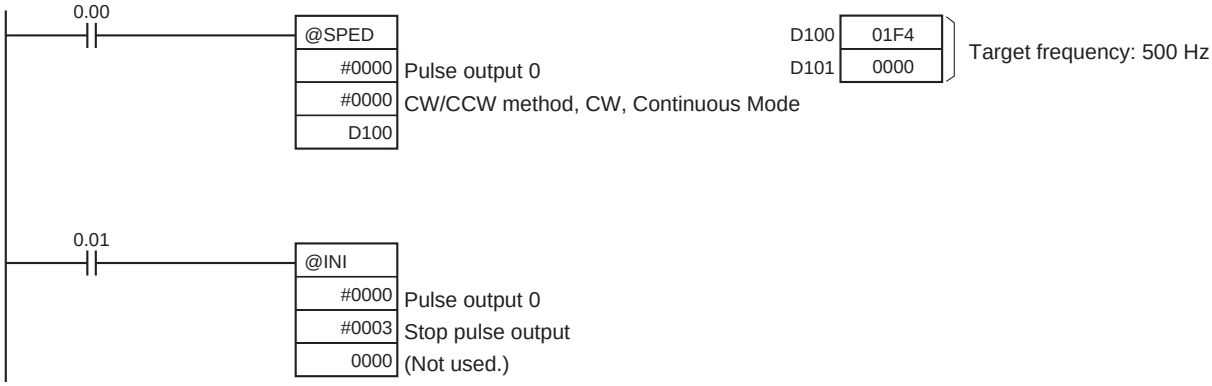
If C is 0003 hex, INI(880) immediately stops pulse output for the specified port. If this instruction is executed when pulse output is already stopped, then the pulse amount setting will be cleared.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, C, or NV is exceeded. ON if the combination of P and C is not allowed. ON if a comparison table has not been registered but starting comparison is specified. ON if a new PV is specified for a port that is currently outputting pulses. ON if changing the PV of a high-speed counter is specified for a port that is not specified for a high-speed counter. ON if a value that is out of range is specified as the PV for an interrupt input in counter mode. ON if INI(880) is executed in an interrupt task for a high-speed counter and an interrupt occurs when CTBL(882) is executed. ON if executed for a port not set for an interrupt input in counter mode.

Example

When CIO 0.00 turns ON in the following example, SPED(885) starts outputting pulses from pulse output 0 in Continuous Mode at 500 Hz. When CIO 0.01 turns ON, pulse output is stopped by INI(880).



3-20-2 HIGH-SPEED COUNTER PV READ: PRV(881)

Purpose

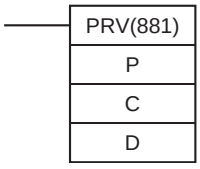
PRV(881) reads the following data on the built-in I/O.

- PVs: High-speed counter PV, pulse output PV, interrupt input PV in counter mode.
- The following status information.

Status type	Contents
Pulse output status	Pulse Output Status Flag PV Underflow/Overflow Flag Pulse Output Amount Set Flag Pulse Output Completed Flag Pulse Output Flag No-origin Flag At Origin Flag Pulse Output Stopped Error Flag
High-speed counter input status	Comparison In-progress Flag PV Underflow/Overflow Flag
PWM(891) output status	Pulse Output In-progress Flag

- Range comparison results
- Pulse output frequency of pulse output 0 to pulse output 3
- High-speed counter frequency for high-speed counter input 0.

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
Immediate Refreshing Specification		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
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3
0010 hex	High-speed counter 0
0011 hex	High-speed counter 1
0012 hex	High-speed counter 2
0013 hex	High-speed counter 3
0100 hex	Interrupt input 0 in counter mode
0101 hex	Interrupt input 1 in counter mode
0102 hex	Interrupt input 2 in counter mode
0103 hex	Interrupt input 3 in counter mode
0104 hex	Interrupt input 4 in counter mode
0105 hex	Interrupt input 5 in counter mode
0106 hex	Interrupt input 6 in counter mode
0107 hex	Interrupt input 7 in counter mode
1000 hex	PWM output 0
1001 hex	PWM output 1

Note Input interrupts 6 and 7 cannot be used on Y CPU Units.

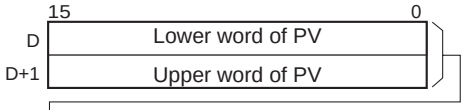
C: Control Data

The function of INI(880) is determined by the control data, C.

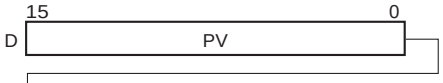
C	PRV(881) function	Variations
0000 hex	Reads the PV.	---
0001 hex	Reads status.	---
0002 hex	Reads range comparison results.	---
00□3 hex	P = 0000 or 0001: Reads the output frequency of pulse output 0 or pulse output 1. P = 0010: Reads the frequency of high-speed counter input 0.	C = 0003 hex: Standard operation C = 0013 hex: 10-ms sampling method for high frequency C = 0023 hex: 100-ms sampling method for high frequency C = 0033 hex: 1-s sampling method for high frequency

D: First Destination Word

The PV is output to D or to D and D+1.



2-word PV
Pulse output PV, high-speed counter input PV,
high-speed counter input frequency for high-speed counter input 0



1-word PV
Interrupt input PV in counter mode, status, range comparison results

Operand Specifications

Area	P	C	D
CIO Area	---	---	CIO 0 to CIO 6142
Work Area	---	---	W0 to W510
Holding Bit Area	---	---	H0 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D0 to D32766
Indirect DM addresses in binary	---	---	@ D0 to @ D32766
Indirect DM addresses in BCD	---	---	*D0 to *D32766
Constants	See description of operand.	See description of operand.	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

PRV(881) reads the data specified in C for the port specified in P. The possible combinations of data and ports are shown in the following table.

P: Port specifier	C: Control data			
	0000 hex: Read PV	0001 hex: Read status	0002 hex: Read range comparison results	0003 hex: Pulse output read high-speed counter frequency
0000 to 0003 hex: Pulse output	OK	OK	Not allowed.	OK
0010 to 0013 hex: High-speed counter input	OK	OK	OK	OK (high-speed counter 0 only)

P: Port specifier	C: Control data			
	0000 hex: Read PV	**0001 hex: Read status**	**0002 hex: Read range comparison results**	**0003 hex: Pulse output read high-speed counter frequency**
0100 to 0107 hex: Interrupt input in counter mode	OK	Not allowed.	Not allowed.	Not allowed.
1000 or 1001 hex: PWM output	Not allowed.	OK	Not allowed.	Not allowed.

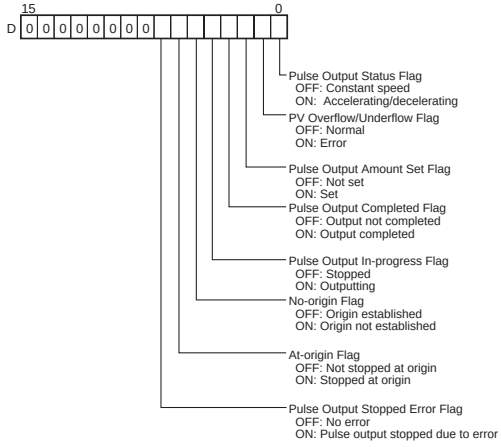
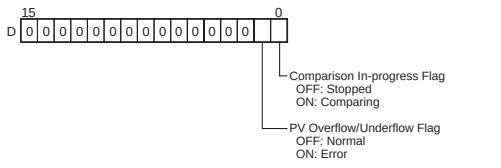
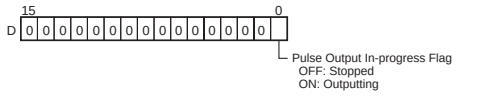
■ **Reading a PV (C = 0000 hex)**

If C is 0000 hex, PRV(881) reads a PV as shown in the following table.

Port and mode		Operation	Setting range
Pulse output (P = 0000 to 0003 hex)		The present value of the pulse output is stored in D and D+1.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
High-speed counter input (P = 0010 to 0013 hex)	Linear Mode	The present value of the high-speed counter is stored in D and D+1.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
	Ring Mode		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
Interrupt inputs in counter mode (P = 0100 to 0107 hex)		The present value of the interrupt input is stored in D.	0000 to FFFF hex (0 to 65,535)

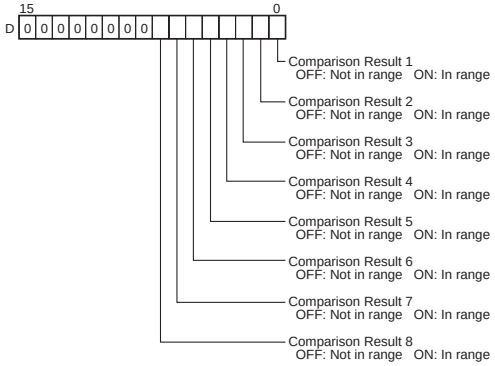
■ **Reading Status (C = 0001 hex)**

If C is 0001 hex, PRV(881) reads status as shown in the following table.

Port and mode	Operation	Results of reading
Pulse output	The pulse output status is stored in D.	
High-speed counter input	The high-speed counter status is stored in D.	
PWM output	The PWM output is stored in D.	

■ **Reading the Results of Range Comparison (C = 0002 hex)**

If C is 0002 hex, PRV(881) reads the results of range comparison and stores it in D as shown in the following diagram.



■ **Reading Pulse Output or High-speed Counter Frequency (C = 00□3 hex)**

If C is 00□3 hex, PRV(881) reads the frequency being output from pulse output 0 to 3 or the frequency being input to high-speed counter 0 and stores it in D and D+1.

Read Frequency Ranges

Value of P	CPU Unit type and unit version	Pulse output/ Counter input method	Conversion result
0000 to 0003 hex	Y CPU Unit, any unit version	Pulse output 1 or 2	0000 0000 to 000F 4240 hex (0 to 1,000,000)
		Pulse output 2 or 3	0000 0000 to 0001 86A0 hex (0 to 100,000)
	X/XA CPU Unit, any unit version	Pulse output 0 or 1	0000 0000 to 0001 86A0 hex (0 to 100,000)
	X/XA CPU Unit, unit version 1.1 or later	Pulse output 2 or 3	
	X/XA CPU Unit, unit version 1.0	Pulse output 2 or 3	0000 0000 to 0000 7530 hex (0 to 30,000)
0010 hex	X/XA CPU Unit, any version	Any counter input method other than 4× differential phase mode	0000 0000 to 0001 86A0 hex (0 to 100,000) **Note** If a frequency higher than 100 kHz has been input, the output will remain at the maximum value of 0001 86A0 hex.
		4× differential phase mode	0000 0000 to 0003 0D40 hex (0 to 200,000) **Note** If a frequency higher than 200 kHz has been input, the output will remain at the maximum value of 0003 0D40 hex.
	Y CPU Unit, any version	Any counter input method other than 4× differential phase mode	0000 0000 to 000F 4240 hex (0 to 1,000,000) **Note** If a frequency higher than 1 MHz has been input, the output will remain at the maximum value of 000F 4240 hex.
		4× differential phase mode	0000 0000 to 001E 8480 hex (0 to 2,000,000) **Note** If a frequency higher than 2 MHz has been input, the output will remain at the maximum value of 001E 8480 hex.

Pulse Frequency Calculation Methods

There are two ways to calculate the frequency of pulses output from pulse output 0 to 3 or pulses input to high-speed counter 0.

1. Standard Calculation Method (Earlier Method)

   The count is calculated by counting each pulse regardless of the frequency. At high frequencies, the rising or falling edges of some pulses will be corrupted, resulting in errors (roughly 1% error max. at 100 kHz and the maximum error at 1 MHz).

2. High-frequency Calculation Method

   In this case, the counting method is switched at high and low frequencies.

   - High-frequency counting

     At high frequencies (above 1 kHz), the function counts the number of pulses within a fixed interval (the sampling time) and calculates the frequency from that count. One of the following three sampling times can be selected by setting the rightmost two digits of C.

Sampling time	Value of C	Description
10 ms	0013 hex	Counts the number of pulses every 10 ms. The error is 0.1% max. at 100 kHz. The maximum error will occur at 1 kHz (10%).
100 ms	0023 hex	Counts the number of pulses every 100 ms. The error is 0.01% max. at 100 kHz. The maximum error will occur at 1 kHz (1%).
1 s	0033 hex	Counts the number of pulses every 1 s. The error is 0.001% max. at 100 kHz. The maximum error will occur at 1 kHz (0.1%).

   - Low-frequency counting

     At frequencies below 1 kHz, the Standard Calculation Method is used, regardless of the sampling time setting.

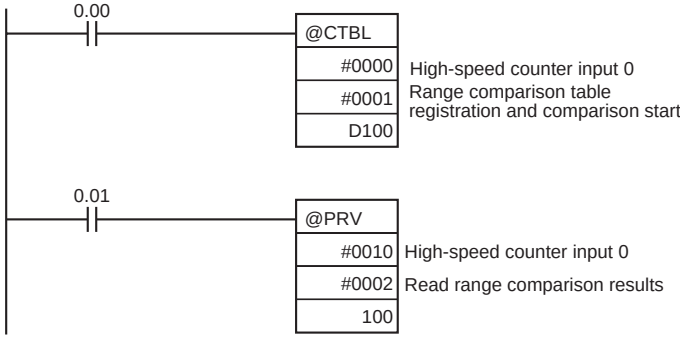
Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P or C is exceeded. ON if the combination of P and C is not allowed. ON if reading range comparison results is specified even though range comparison is not being executed. ON if reading the output frequency is specified for anything except for high-speed counter 0. ON if specified for a port not set for a high-speed counter. ON if executed for a port not set for an interrupt input in counter mode.

Examples

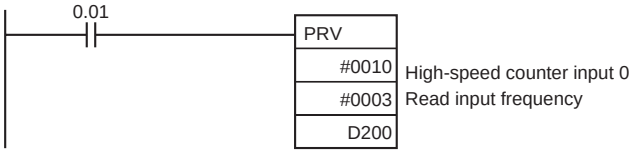
Example 1

When CIO 0.00 turns ON in the following programming example, CTBL(882) registers a range comparison table for high-speed counter 0 and starts comparison. When CIO 0.01 turns ON, PRV(881) reads the range comparison results at that time and stores them in CIO 100.00.



Example 2

When CIO 0.01 turns ON in the following programming example, PRV(881) reads the frequency of the pulse being input to high-speed counter 0 at that time and stores it as a hexadecimal value in D201 and D200.

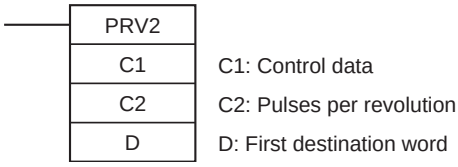


3-20-3 COUNTER FREQUENCY CONVERT: PRV2(883)

Purpose

PRV2(883) reads the pulse frequency input from a high-speed counter and either converts the frequency to a rotational speed or converts the counter PV to the total number of revolutions. The result is output to the destination words as 8-digit hexadecimal. Pulses can be input from high-speed counter 0 only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	PRV2(883)
	Executed Once for Upward Differentiation	@PRV2(883)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

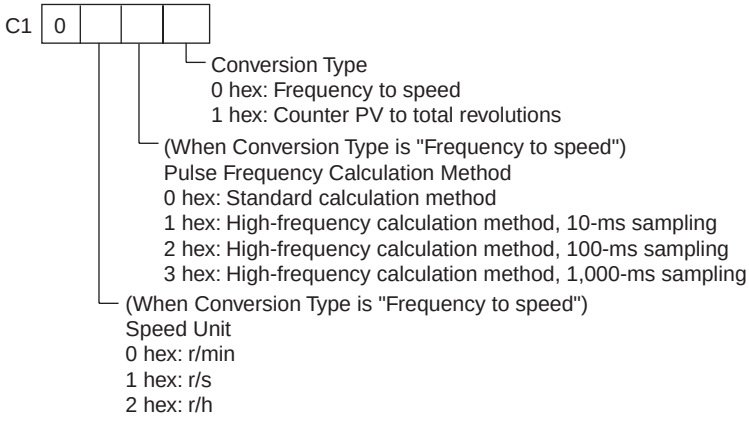
Operands

C1: Control Data

The function of PRV2(883) is determined by the control data, C1.

C1	PRV2(883) function
0□∗0 hex (See note.)	Converts frequency to rotation speed.
0001 hex	Converts counter PV to total number of revolutions.

Note The second digit of C (□) specifies the units and the third digit (∗) specifies the frequency calculation method.

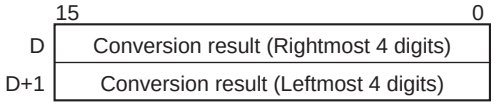


C2: Pulses per Revolution

Specifies the number of pulses per revolution (0001 to FFFF hex).

D: First Destination Word

The PV is output to D or to D and D+1.



Operand Specifications

Area	C1	C2	D
CIO Area	---	CIO 0 to CIO 6143	CIO 0 to CIO 6142
Work Area	---	W0 to W511	W0 to W510
Holding Bit Area	---	H0 to H511	H0 to H510
Auxiliary Bit Area	---	A448 to A959	A448 to A958
Timer Area	---	T0000 to T4095	T0000 to T4094
Counter Area	---	C0000 to C4095	C0000 to C4094
DM Area	---	D0 to D32767	D0 to D32766
Indirect DM addresses in binary	---	@ D0 to @ D32767	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767	*D0 to *D32767
Constants	See description of operand.	---	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

PRV2(883) converts the pulse frequency input from high-speed counter 0, according to the conversion method specified in C1 and the pulses/revolution coefficient specified in C2, and outputs the result to D and D+1.

Select one of the following conversion methods by setting C1 to 0000 hex or 0001 hex.

Converting Frequency to Rotation Speed (C1 = 0□∗0 hex)

If C1 is 0□∗0 hex, PRV2(883) calculates the rotation speed (r/min) from the frequency data and pulses/revolution setting. The second digit of C (□) specifies the units and the third digit (∗) specifies the frequency calculation method.

1. Rotation Speed Units
   - Rotation Speed Units = r/min

     When the second digit of C (□) is 0, PRV2(883) calculates the rotation speed in r/min from the frequency data and pulses/revolution setting.

     Rotation speed (r/min) = (Frequency ÷ Pulses/revolution) × 60
   - Rotation Speed Units = r/s

     When the second digit of C (□) is 1, PRV2(883) calculates the rotation speed in r/s from the frequency data and pulses/revolution setting.

     Rotation speed (r/s) = Frequency ÷ Pulses/revolution
   - Rotation Speed Units = r/hr

     When the second digit of C (□) is 2, PRV2(883) calculates the rotation speed in r/hr from the frequency data and pulses/revolution setting.

     Rotation speed (r/hr) = (Frequency ÷ Pulses/revolution) × 60 × 60
   - Range of Conversion Results for X/XA CPU Units
     - Counter input method: Any method besides 4× differential phase mode
       Conversion result = 0000 0000 to 0001 86A0 hex (0 to 100,000)
     - Counter input method: 4× differential phase mode
       Conversion result = 0000 0000 to 0003 0D40 hex (0 to 200,000)
       (If a frequency higher than 200 kHz has been input, the output will remain at the maximum value of 0003 0D40 hex.)
   - Range of Conversion Results for Y CPU Units
     - Counter input method: Any method besides 4× differential phase mode
       Conversion result = 0000 0000 to 000F 4240 hex (0 to 1,000,000)
     - Counter input method: 4× differential phase mode
       Conversion result = 0000 0000 to 001E 8480 hex (0 to 2,000,000)
       (If a frequency higher than 2 MHz has been input, the output will remain at the maximum value of 001E 8480 hex.)
2. Frequency Calculation Method

   There are two ways to calculate the frequency of pulses input to high-speed counter 0.

   a. Standard Calculation Method (C1 = 0□00)

      The count is calculated by counting each pulse regardless of the frequency. At high frequencies, the rising or falling edges of some pulses will be corrupted, resulting in errors (about 1% error max. at 100 kHz).

   b. High-frequency Calculation Method

      In this case, the counting method is switched at high and low frequencies.

      - High-frequency counting (C1 = 0□10, 0□20, or 0□30)

At high frequencies (above 1 kHz), the function counts the number of pulses within a fixed interval (the sampling time) and calculates the frequency from that count. One of the following three sampling times can be selected by the third digit of C1.

Sampling time	Value of C1	Description
10 ms	0□10 hex	Counts the number of pulses every 10 ms. The error is 10% max. at 1 kHz.
100 ms	0□20 hex	Counts the number of pulses every 100 ms. The error is 1% max. at 1 kHz.
1 s	0□30 hex	Counts the number of pulses every 1 s. The error is 0.1% max. at 1 kHz.

- Low-frequency counting

  At frequencies below 1 kHz, the Standard Calculation Method is used, regardless of the sampling time setting.

Converting Counter PV to Total Number of Revolutions (C1 = 0001 hex)

If C1 is 0001 hex, PRV2(883) calculates the cumulative number of revolutions from the counter PV and pulses/revolution setting.

Conversion result = Counter PV ÷ Pulses/revolution

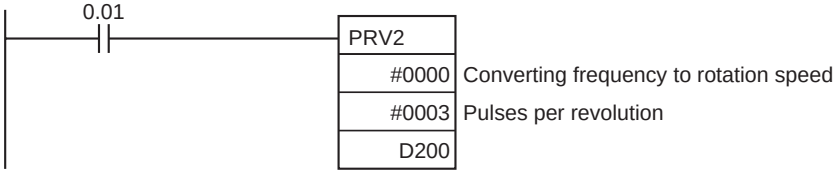
Flags

Name	Label	Operation
Error Flag	ER	ON if high-speed counter 0 is disabled in the settings. ON if C1 is not in the specified range (0000 or 0001). ON if the pulses/revolution setting in C2 is 0000.
Overflow Flag	OF	ON if the maximum value is exceeded for a Y CPU Unit.

Examples

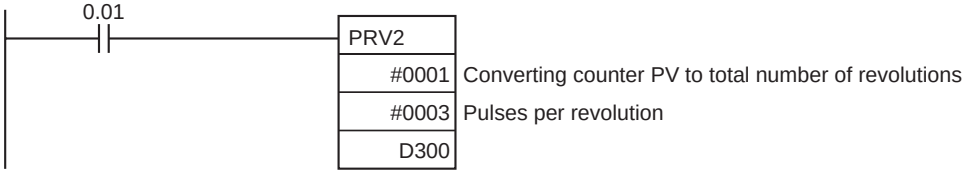
Example 1

When CIO 0.01 is ON in the following programming example, PRV2(883) reads the present pulse frequency at high-speed counter 0, converts that value to rotation speed (r/min), and outputs the hexadecimal result to D201 and D200.



Example 2

When CIO 0.01 is ON in the following programming example, PRV2(883) reads the counter PV, converts that value to number of revolutions, and outputs the hexadecimal result to D301 and D300.

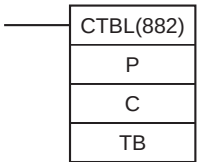


3-20-4 REGISTER COMPARISON TABLE: CTBL(882)

Purpose

CTBL(882) is used to register a comparison table and perform comparisons for a high-speed counter PV. Either target value or range comparisons are possible. An interrupt task is executed when a specified condition is met.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	CTBL(882)
	Executed Once for Upward Differentiation	@CTBL(882)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		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
0000 hex	High-speed counter 0
0001 hex	High-speed counter 1
0002 hex	High-speed counter 2
0003 hex	High-speed counter 3

C: Control Data

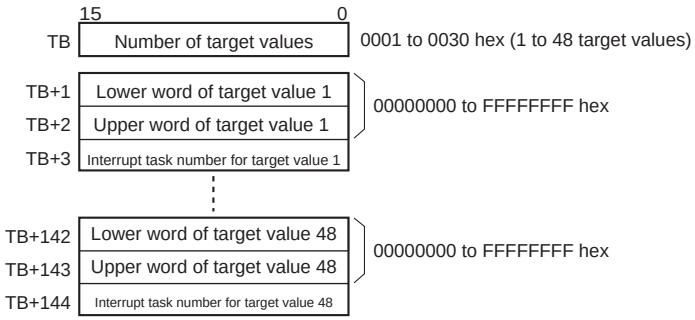
The function of CTBL(882) is determined by the control data, C, as shown in the following table.

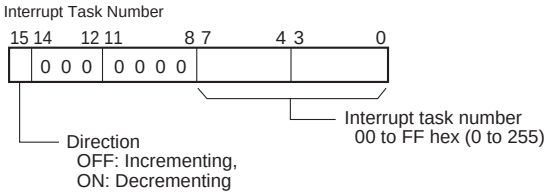
C	CTBL(882) function
0000 hex	Registers a target value comparison table and starts comparison.
0001 hex	Registers a range comparison table and performs one comparison.
0002 hex	Registers a target value comparison table. Comparison is started with INI(880).
0003 hex	Registers a range comparison table. Comparison is started with INI(880).

TB: First Table Comparison Word

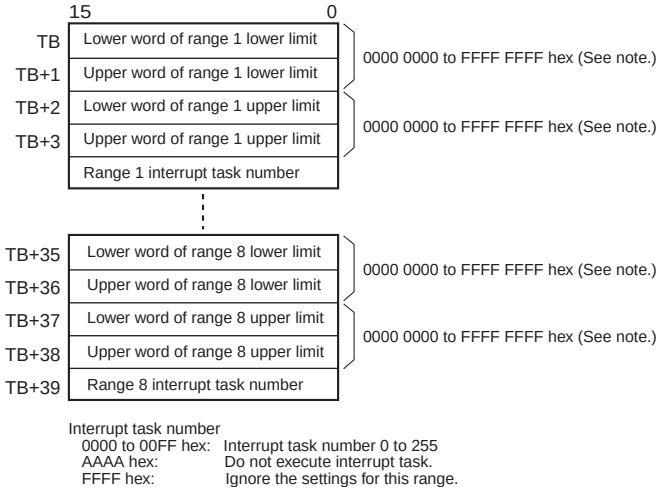
TB is the first word of the comparison table. The structure of the comparison table depends on the type of comparison being performed.

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.





For range comparison, the comparison table always contains eight ranges. The table is 40 words long, as shown below. If it is not necessary to set eight ranges, set the interrupt task number to FFFF hex for all unused ranges.



Note Always set the upper limit greater than or equal to the lower limit for any one range.

Operand Specifications

Area	P	C	TB
CIO Area	---	---	CIO 0 to CIO 6143
Work Area	---	---	W0 to W511
Holding Bit Area	---	---	H0 to H511
Auxiliary Bit Area	---	---	A448 to A959
Timer Area	---	---	T0000 to T4095
Counter Area	---	---	C0000 to C4095
DM Area	---	---	D0 to D32767
Indirect DM addresses in binary	---	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	---	*D0 to *D32767

Area	P	C	TB
Constants	See description of operand.	See description of operand.	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

CTBL(882) registers a comparison table or registers and comparison table and starts comparison for the port specified in P and the method specified in C. Once a comparison table is registered, it is valid until a different table is registered or until the CPU Unit 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 version (@CTBL(882)) of the instruction or an execution condition that is turned ON only for one scan.

Note If an interrupt task that has not been registered is specified, a fatal program error will occur the first time an interrupt is generated.

■ Registering a Comparison Table (C = 0002 or 0003 hex)

If C is set to 0002 or 0003 hex, a comparison table will be registered, but comparison will not be started. Comparison is started with INI(880).

■ Registering a Comparison Table and Starting Comparison (C = 0000 or 0001 hex)

If C is set to 0000 or 0001 hex, a comparison table will be registered, and comparison will be started.

■ Stopping Comparison

Comparison is stopped with INI(880). It makes no difference what instruction was used to start comparison.

■ Target Value Comparison

The corresponding interrupt task is called and executed when the PV matches a target value.

- The same interrupt task number can be specified for more than one target value.
- The direction can be set to specify whether the target value is valid when the PV is being incremented or decremented. If bit 15 in the word used to specify the interrupt task number for the range is OFF, the PV will be compared to the target value only when the PV is being incremented, and if bit 00 is ON, only when the PV is being decremented.
- The comparison table can contain up to 48 target values, and the number of target values is specified in TB (i.e., the length of the table depends on the number of target values that is specified).
- Comparisons are performed for all target values registered in the table.

Note (1) An error will occur if the same target value with the same comparison direction is registered more than once in the same table.

(2) If the high-speed counter is set for incremental pulse mode, an error will occur if decrementing is set in the table as the direction for comparison.

(3) If the count direction changes while the PV equals a target value that was reached in the direction opposite to that set as the comparison direction, the comparison condition for that target value will not be met. Do not set target values at peak and bottom values of the count value.

■ Range Comparison

The corresponding interrupt task is called and executed when the PV enters a set range.

- The same interrupt task number can be specified for more than one target value.
- The range comparison table contains 8 ranges, each of which is defined by a lower limit and an upper limit. If a range is not to be used, set the interrupt task number to FFFF hex to disable the range.
- The interrupt task is executed only once when the PV enters the range.
- If the PV is within more than one range when the comparison is made, the interrupt task for the range closest to the beginning of the table will be given priority and other interrupt tasks will be executed in following cycles.
- If there is no reason to execute an interrupt task, specify AAAA hex as the interrupt task number. The range comparison results can be read with PRV(881) or using the Range Comparison In-progress Flags.

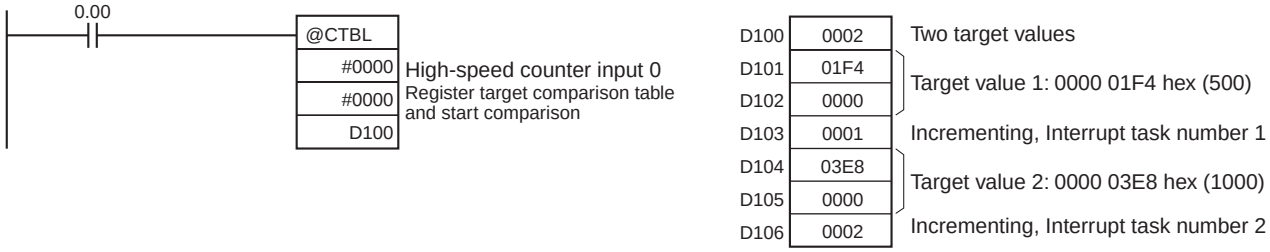
Note An error will occur if the upper limit is less than the lower limit for any one range.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P or C is exceeded. ON if the number of target values specified for target value comparison is set to 0. ON if the number of target values specified for target value comparison exceeds 48. ON if the same target value is specified more than once in the same comparison direction for target comparison. ON if the upper value is less than the lower value for any range. ON if the set values for all ranges are disabled during a range comparison. ON if the high-speed counter is set for incremental pulse mode and decrementing is set in the table as the direction for comparison. ON if an instruction is executed when the high-speed counter is set to Ring Mode and the specified value exceeds the maximum ring value. ON if specified for a port not set for a high-speed counter. ON if executed for a different comparison method while comparison is already in progress.

Example

When CIO 0.00 turns ON in the following programming example, CTBL(882) registers a target value comparison table and starts comparison for high-speed counter 0. 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 1000, it equals target value 2 and interrupt task 2 is executed.



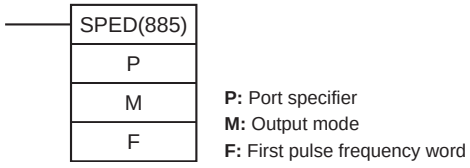
3-20-5 SPEED OUTPUT: SPED(885)

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 output frequency, creating stepwise changes in the speed.

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

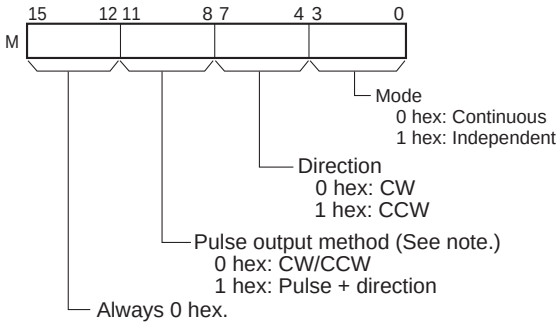
P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3

M: Output Mode

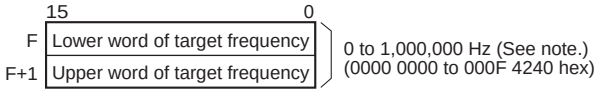
The value of M determines the output mode.



Note: Use the same pulse output method when using both pulse outputs 0 and 1.

F: First Pulse Frequency Word

The value of F and F+1 sets the pulse frequency in Hz.



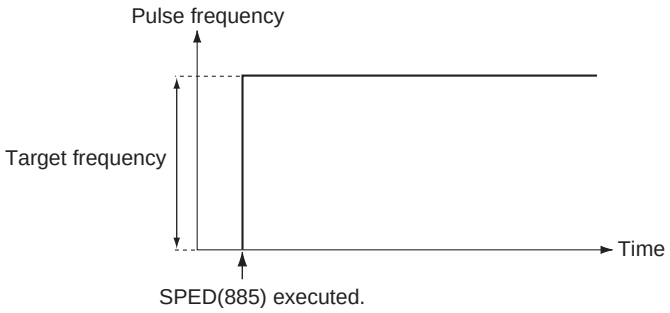
Note The maximum frequency that can be specified depends on the model and pulse output support. Refer to the *CP1H Operation Manual.*

Operand Specifications

Area	P	M	F
CIO Area	---	---	CIO 0 to CIO 6142
Work Area	---	---	W0 to W510
Holding Bit Area	---	---	H0 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D0 to D32766
Indirect DM addresses in binary	---	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	---	*D0 to *D32767
Constants	See description of operand.	See description of operand.	See description of operand.
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

SPED(885) starts pulse output on the port specified in P using the method specified in M at the frequency specified in F. Pulse output will be started each time SPED(885) is executed. It is thus normally sufficient to use the differentiated version (@SPED(885)) of the instruction or an execution condition that is turned ON only for one scan.



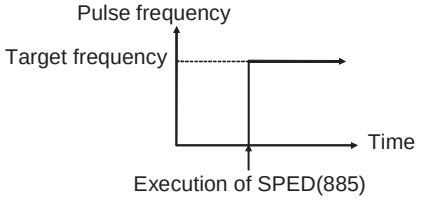
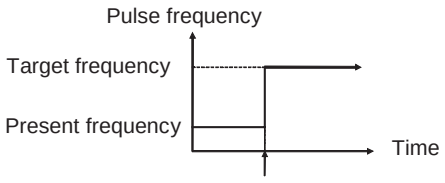
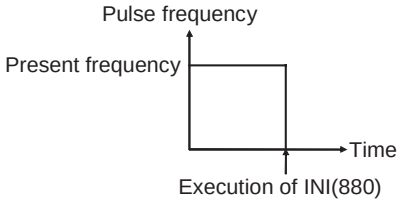
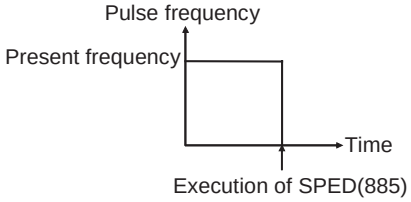
In independent mode, pulse output will stop automatically when the number of pulses set with PULS(886) in advance have been output. In continuous mode, pulse output will continue until stopped from the program.

An error will occur if the mode is changed between independent and continuous mode while pulses are being output.

■ Continuous Mode Speed Control

When continuous mode operation is started, pulse output will be continued until it is stopped from the program.

Note Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified speed	Changing the speed (frequency) in one step		Outputs pulses at a specified frequency.	SPED(885) (Continuous)
Changing settings	To change speed in one step	Changing the speed during operation		Changes the frequency (higher or lower) of the pulse output in one step.	SPED(885) (Continuous) ↓ SPED(885) (Continuous)
Stopping pulse output	Stop pulse output	Immediate stop		Stops the pulse output immediately.	SPED(885) (Continuous) ↓ INI(880)
	Stop pulse output	Immediate stop		Stops the pulse output immediately.	SPED(885) (Continuous) ↓ SPED(885) (Continuous, Target frequency of 0 Hz)

■ Independent Mode Positioning

When independent mode operation is started, pulse output will be continued until the specified number of pulses has been output.

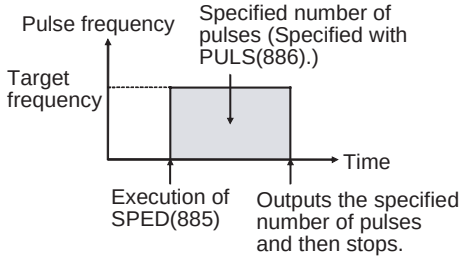
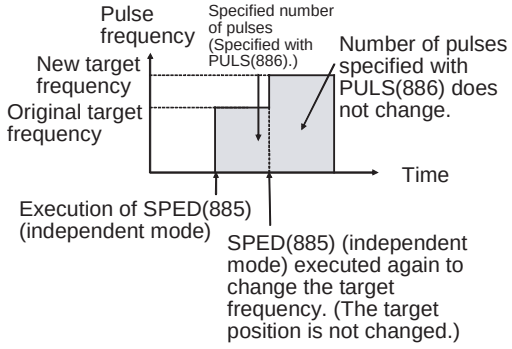
Note

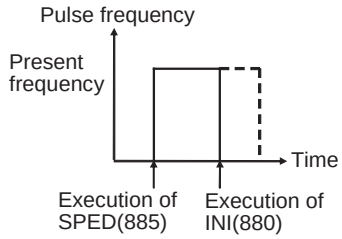
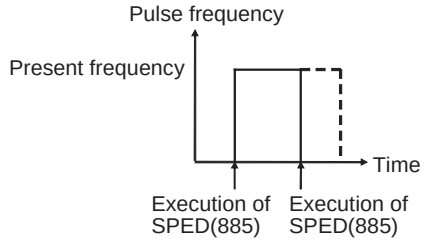
(1) Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

(2) The number of output pulses must be set each time output is restarted.

(3) The number of output pulses must be set in advance with PULS(881). Pulses will not be output for SPED(885) if PULS(881) is not executed first.

(4) The direction set in the SPED(885) operand will be ignored if the number of pulses is set with PULS(881) as an absolute value.

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified speed	Positioning without acceleration or deceleration		Starts outputting pulses at the specified frequency and stops immediately when the specified number of pulses has been output. **Note** The target position (specified number of pulses) cannot be changed during positioning.	PULS(886) ↓ SPED(885) (Independent)
Changing settings	To change speed in one step	Changing the speed in one step during operation		SPED(885) can be executed during positioning to change (raise or lower) the pulse output frequency in one step. The target position (specified number of pulses) is not changed.	PULS(886) ↓ SPED(885) (Independent) ↓ SPED(885) (Independent)

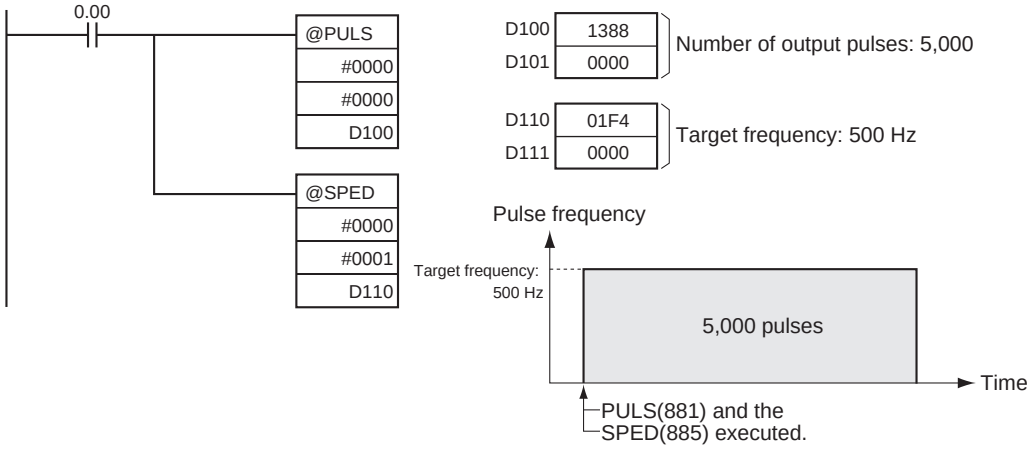
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output	To stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses setting.	PULS(886) ↓ SPED(885) (Independent) ↓ INI(880)
					PLS2(887) ↓ INI(880)
	Stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses setting.	PULS(886) ↓ SPED(885) (Independent) ↓ SPED(885), (Independent, Target frequency of 0 Hz)

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, M, or F is exceeded. ON if PLS2(887) or ORG(889) is already being executed to control pulse output for the specified port. ON if SPED(885) or INI(880) is used to change the mode between continuous and independent output during pulse output. ON if SPED(885) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if SPEC(885) is executed in independent mode with an absolute number of pulses and the origin has not been established.

Example

When CIO 0.00 turns ON in the following programming example, PULS(886) sets the number of output pulses for pulse output 0. An absolute 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.

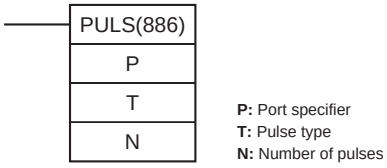


3-20-6 SET PULSES: PULS(886)

Purpose

PULS(886) is used to set the pulse output amount (number of output pulses) for pulse outputs that are started later in the program using SPED(885) or ACC(888) in independent mode.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	PULS(886)
	Executed Once for Upward Differentiation	@PULS(886)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		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. The parameters set in D and N will apply to the next SPED(885) or ACC(888) instruction in which the same port output location is specified.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3

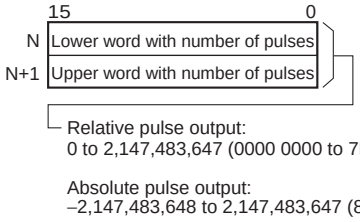
T: Pulse Type

T specifies the type of pulses that are output as follows:

T	Pulse type
0000 hex	Relative
0001 hex	Absolute

N and N+1: Number of Pulses

N and N+1 specify the number of pulses for relative pulse output or the absolute target position for absolute pulse in 8-digit hexadecimal.



The actual number of movement pulses that will be output are as follows:

For relative pulse output, the number of movement pulses = the set number of pulses. For absolute pulse output, the number of movement pulses = the set number of pulses – the PV.

Operand Specifications

Area	P	T	N
CIO Area	---	---	CIO 0 to CIO 6142
Work Area	---	---	W0 to W510
Holding Bit Area	---	---	H0 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D0 to D32766
Indirect DM addresses in binary	---	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	---	*D0 to *D32767
Constants	See description of operand.	See description of operand.	See description of operand.
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 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.

Flags

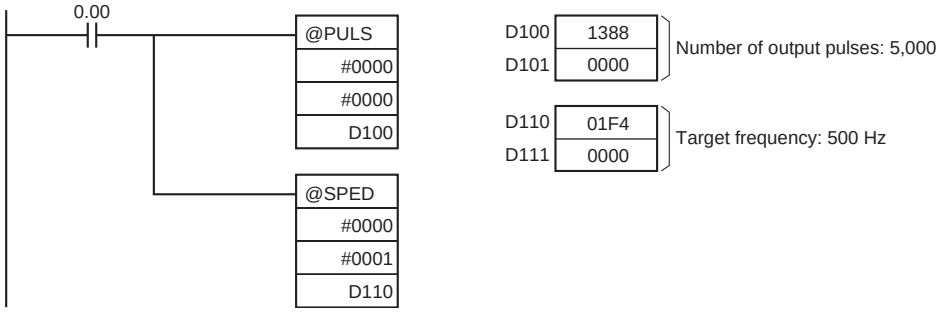
Name	Label	Operation
Error Flag	ER	ON if the specified range for P, T, or N is exceeded. ON if PULS(886) is executed for a port that is already outputting pulses. ON if PULS(886) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task.

Precautions

- An error will occur if PULS(886) is executed when pulses are already being output. Use the differentiated version (@PULS(886)) of the instruction or an execution condition that is turned ON only for one scan to prevent this.
- The calculated number of pulses output for PULS(886) will not change even if INI(880) is used to change the PV of the pulse output.
- The direction set for SPED(885) or ACC(888) will be ignored if the number of pulses is set with PULS(881) as an absolute value.
- It is possible to move outside of the range of the PV of the pulse output amount (–2,147,483,648 to 2,147,483,647).

Example

When CIO 0.00 turns ON in the following programming example, PULS(886) sets the number of output pulses for pulse output 0. An absolute 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.



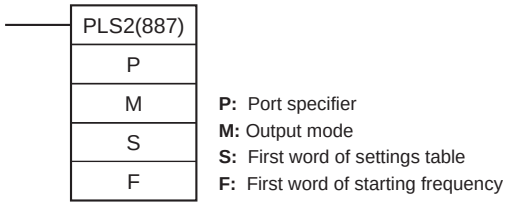
3-20-7 PULSE OUTPUT: PLS2(887)

Purpose

PLS2(887) outputs a specified number of pulses to the specified port. Pulse output starts at a specified startup frequency, accelerates to the target frequency at a specified acceleration rate, decelerates at the specified deceleration rate, and stops at approximately the same frequency as the startup frequency. Only independent mode positioning is supported.

PLS2(887) can also be executed during pulse output to change the number of output pulses, target frequency, acceleration rate, or deceleration rate. PLS2(887) can thus be used for sloped speed changes with different acceleration and deceleration rates, target position changes, target and speed changes, or direction changes.

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

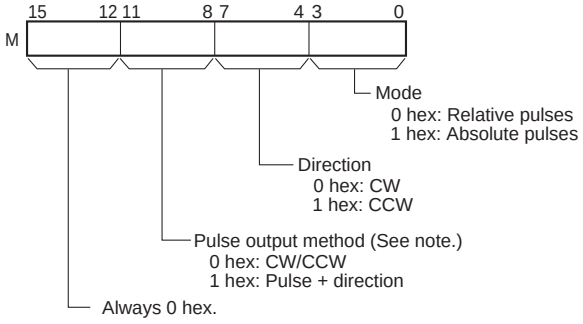
P: Port Specifier

The port specifier indicates the port.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3

M: Output Mode

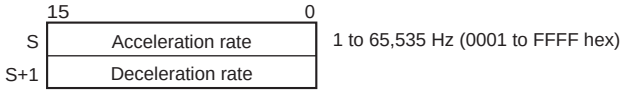
The content of M specifies the parameters for the pulse output as follows:



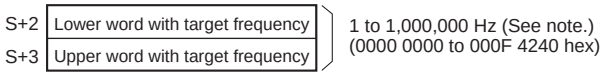
Note: Use the same pulse output method when using both pulse outputs 0 and 1.

S: First Word of Settings Table

The contents of S to S+5 control the pulse output as shown in the following diagrams.

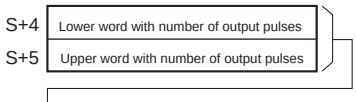


Specify the increase or decrease in the frequency per pulse control period (4 ms).



Specify the frequency after acceleration in Hz.

Note: The maximum frequency that can be specified depends on the model and pulse output support. Refer to the *CP1H Operation Manual.*



Relative pulse output: 0 to 2,147,483,647
(0000 0000 to 7FFF FFFF hex)

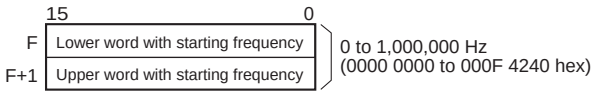
Absolute pulse output: -2,147,483,648 to 2,147,483,647
(8000 0000 to 7FFF FFFF hex)

The actual number of movement pulses that will be output are as follows:

For relative pulse output, the number of movement pulses = the set number of pulses. For absolute pulse output, the number of movement pulses = the set number of pulses – the PV.

F: First Word of Starting Frequency

The starting frequency is given in F and F+1.



Specify the starting frequency in Hz.

Operand Specifications

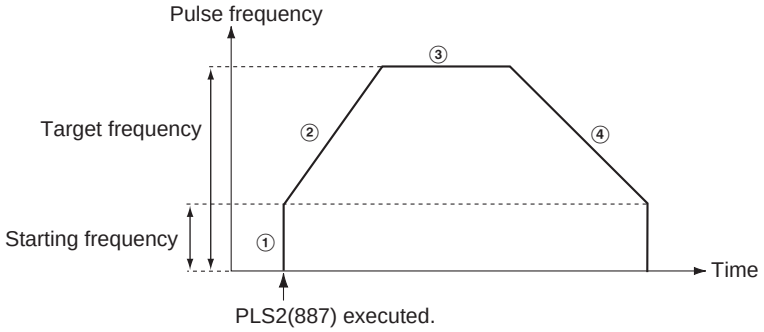
Area	P	M	S	F
CIO Area	---	---	CIO 0 to CIO 6138	CIO 0 to CIO 6142
Work Area	---	---	W0 to W506	W0 to W510
Holding Bit Area	---	---	H0 to H506	H0 to H510
Auxiliary Bit Area	---	---	A448 to A954	A448 to A958
Timer Area	---	---	T0000 to T4090	T0000 to T4094
Counter Area	---	---	C0000 to C4090	C0000 to C4094
DM Area	---	---	D0 to D32762	D0 to D32766
Indirect DM addresses in binary	---	---	@ D0 to @ D32767	@ D0 to @ D32767
Indirect DM addresses in BCD	---	---	*D0 to *D32767	*D0 to *D32767
Constants	See description of operand.	See description of operand.	---	See description of operand.
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	,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

PLS2(887) starts pulse output on the port specified in P using the mode specified in M at the start frequency specified in F (1 in diagram). The frequency is increased every pulse control period (4 ms) at the acceleration rate specified in S until the target frequency specified in S is reached (2 in diagram). When the target frequency has been reached, acceleration is stopped and pulse output continues at a constant speed (3 in diagram).

The deceleration point is calculated from the number of output pulses and deceleration rate set in S and when that point is reached, the frequency is decreased every pulse control period (4 ms) at the deceleration rate specified in S until the starting frequency specified in S is reached, at which point pulse output is stopped (4 in diagram).

Pulse output is started each time PLS2(887) is executed. It is thus normally sufficient to use the differentiated version (@PLS2(887)) of the instruction or an execution condition that is turned ON only for one scan.



PLS2(887) can be used only for positioning.

PLS2(887) can be executed during pulse output for ACC(888) in either independent or continuous mode, and during acceleration, constant speed, or deceleration. (See note.) ACC(888) can also be executed during pulse output for PLS2(887) during acceleration, constant speed, or deceleration.

Note

(1) Executing PLS2(887) during speed control with ACC(888) (continuous mode) with the same target frequency as ACC(888) can be used to achieve interrupt feeding of a fixed distance. Acceleration will not be performed by PLS2(887) for this application, but if the acceleration rate is set to 0, the Error Flag will turn ON and PLS2(887) will not be executed. Always set the acceleration rate to a value other than 0.

(2) The acceleration/deceleration rate can be specified as 1 Hz or higher. There is no upper limit to the acceleration/deceleration time. If the difference between the starting speed and target speed is more than 100 kHz, the acceleration/deceleration rate will be automatically increased.

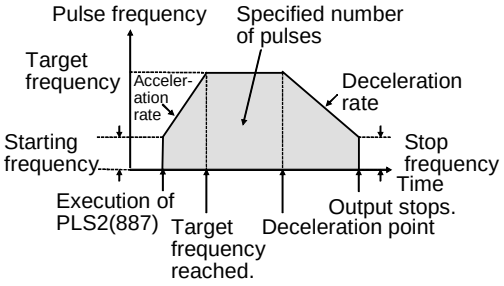
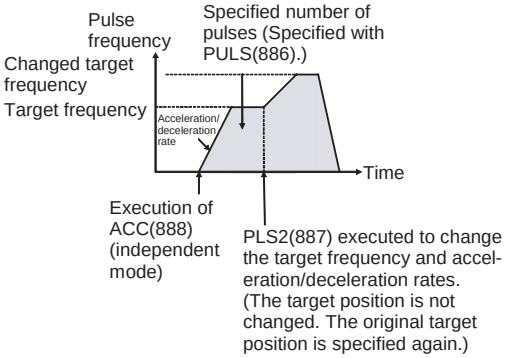
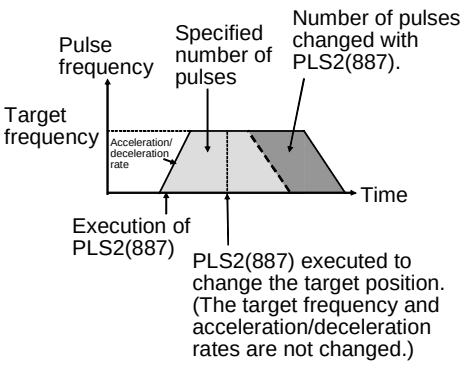
- If the difference between the starting speed and target speed is between 100 and 200 KHz, the acceleration/deceleration rate will be 2 Hz or higher.
- If the difference between the starting speed and target speed is between 200 and 300 KHz, the acceleration/deceleration rate will be 3 Hz or higher.

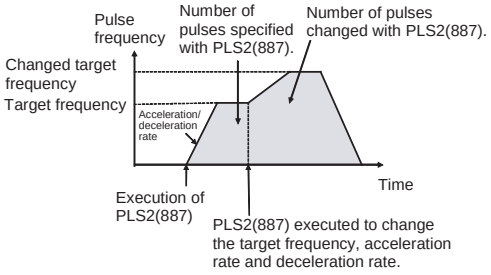
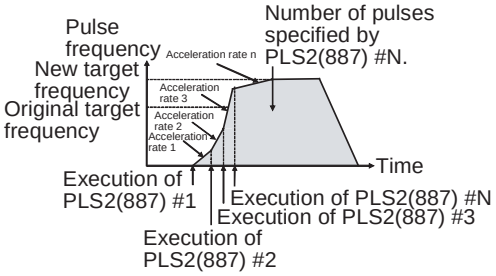
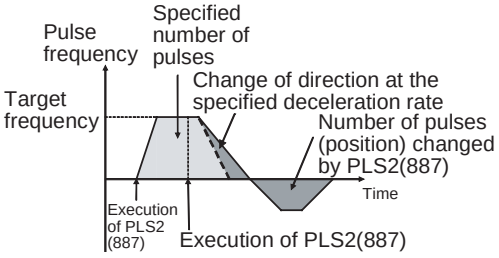
  :

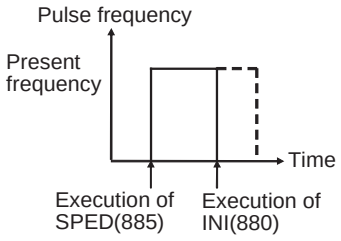
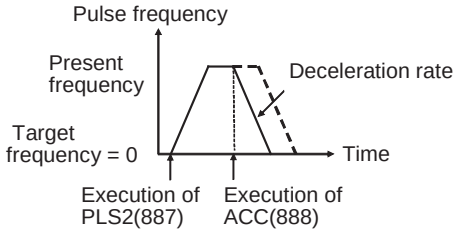
- If the difference between the starting speed and target speed is between 900 and 1,000 KHz, the acceleration/deceleration rate will be 10 Hz or higher.

■ Independent Mode Positioning

Note Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

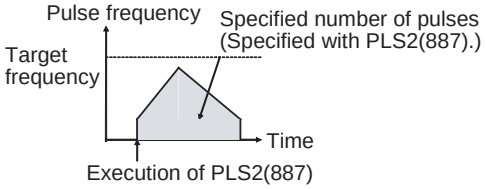
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	Complex trapezoidal control	Positioning with trapezoidal acceleration and deceleration (Separate rates used for acceleration and deceleration; starting speed) The number of pulses can be changed during positioning.		Accelerates and decelerates at a fixed rates. The pulse output is stopped when the specified number of pulses has been output. (See note.) **Note** The target position (specified number of pulses) can be changed during positioning.	PLS2(887)
Changing settings	To change speed smoothly (with unequal acceleration and deceleration rates)	Changing the target speed (frequency) during positioning (different acceleration and deceleration rates)		PLS2(887) can be executed during positioning to change the acceleration rate, deceleration rate, and target frequency. **Note** To prevent the target position from being changed intentionally, the original target position must be specified in absolute coordinates.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
	To change target position	Changing the target position during positioning (multiple start function)		PLS2(887) can be executed during positioning to change the target position (number of pulses), acceleration rate, deceleration rate, and target frequency. **Note** If a constant speed cannot be maintained after changing the settings, an error will occur and the original operation will continue to the original target position.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Changing settings, continued	To change target position and speed smoothly	Changing the target position and target speed (frequency) during positioning (multiple start function)		PLS2(887) can be executed during positioning to change the target position (number of pulses), acceleration rate, deceleration rate, and target frequency. **Note** If a constant speed cannot be maintained after changing the settings, an error will occur and the original operation will continue to the original target position.	PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
		Changing the acceleration and deceleration rates during positioning (multiple start function)		PLS2(887) can be executed during positioning (acceleration or deceleration) to change the acceleration rate or deceleration rate.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
	To change direction	Changing the direction during positioning		PLS2(887) can be executed during positioning with absolute pulse specification to change to absolute pulses and reverse direction.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)

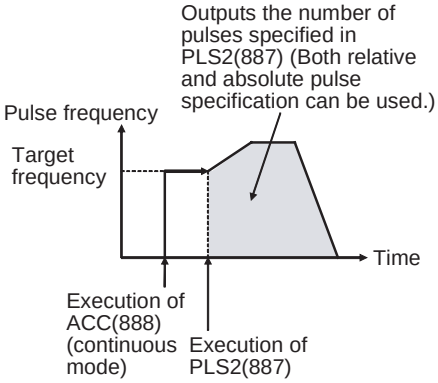
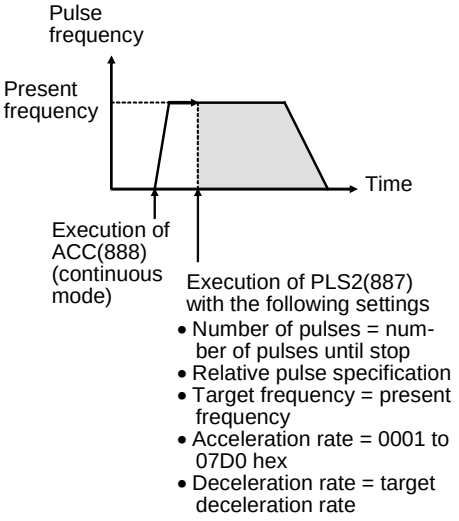
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output	Stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses.	PLS2(887) ↓ INI(880)
	Stop pulse output smoothly. (Number of pulses setting is not preserved.)	Decelerate to a stop		Decelerates the pulse output to a stop.	PLS2(887) ↓ ACC(888) (Independent, target frequency of 0 Hz)

Note Triangular Control

If the specified number of pulses is less than the number required to reach the target frequency and return to zero, the function will automatically reduce the acceleration/deceleration time and perform triangular control (acceleration and deceleration only.) An error will not occur.



■ Switching from Continuous Mode Speed Control to Independent Mode Positioning

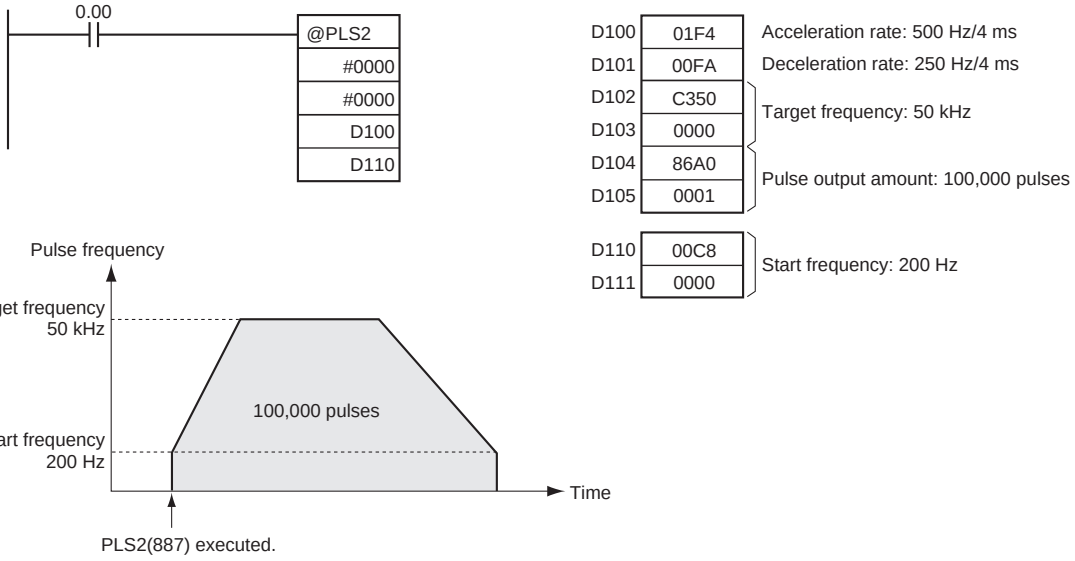
Example application	Frequency changes	Description	Procedure/ instruction
Change from speed control to fixed distance positioning during operation		PLS2(887) can be executed during a speed control operation started with ACC(888) to change to positioning operation.	ACC(888) (Continuous) ↓ PLS2(887)
Fixed distance feed interrupt			

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, M, S, or F is exceeded. ON if PLS2(887) is executed for a port that is already outputting pulses for SPED(885) or ORG(889). ON if PLS2(887) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if PLS2(887) is executed for an absolute pulse output but the origin has not been established.

Example

When CIO 0.00 turns ON in the following programming example, PLS2(887) starts pulse output from pulse output 0 with an absolute pulse specification of 100,000 pulses. Pulse output is accelerated at a rate of 500 Hz every 4 ms starting at 200 Hz until the target speed of 50 kHz is reached. From the deceleration point, the pulse output is decelerated at a rate of 250 Hz every 4 ms starting until the starting speed of at 200 Hz is reached, at which point pulse output is stopped.

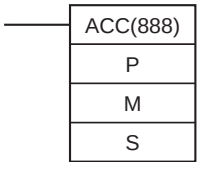


3-20-8 ACCELERATION CONTROL: ACC(888)

Purpose

ACC(888) outputs pulses to the specified output port at the specified frequency using the specified acceleration and deceleration rate. (Acceleration rate is the same as the deceleration rate.) Either independent mode positioning or constant mode speed control is possible. 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.

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
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

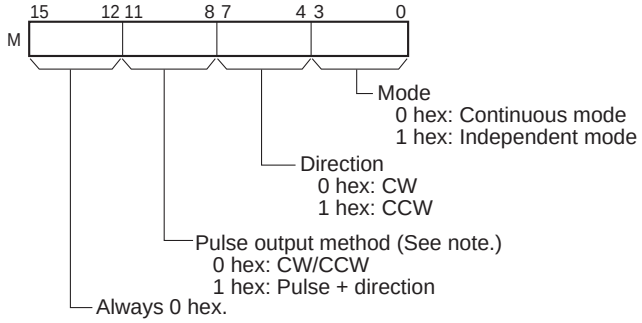
P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3

M: Output Mode

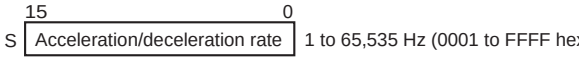
The content of M specifies the parameters for the pulse output as follows:



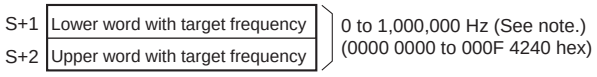
Note: Use the same pulse output method when using both pulse outputs 0 and 1.

S: First Word of Settings Table

The content of S to S+2 controls the pulse output as shown in the following diagrams.



Specify the increase or decrease in the frequency per pulse control period (4 ms).



Specify the frequency after acceleration in Hz.

Note: The maximum frequency that can be specified depends on the model and pulse output support. Refer to the CP1H Operation Manual.

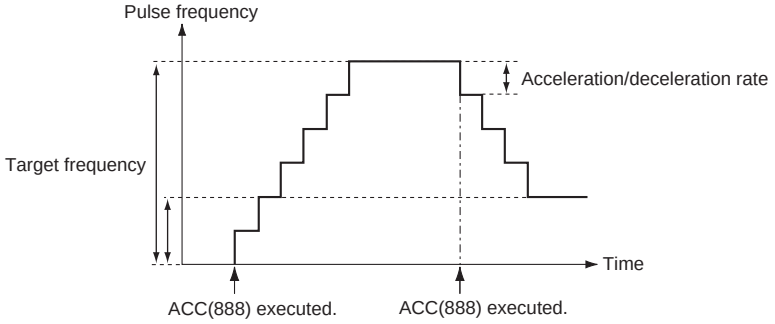
Operand Specifications

Area	P	M	S
CIO Area	---	---	CIO 0 to CIO 6141
Work Area	---	---	W0 to W509
Holding Bit Area	---	---	H0 to H509
Auxiliary Bit Area	---	---	A448 to A957
Timer Area	---	---	T0000 to T4093
Counter Area	---	---	C0000 to C4093
DM Area	---	---	D0 to D32765
Indirect DM addresses in binary	---	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	---	*D0 to *D32767
Constants	See description of operand.	See description of operand.	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ACC(888) starts pulse output on the port specified in P using the mode specified in M using the target frequency and acceleration/deceleration rate specified in S. The frequency is increased every pulse control period (4 ms) at the acceleration rate specified in S until the target frequency specified in S is reached.

Pulse output is started each time ACC(888) is executed. It is thus normally sufficient to use the differentiated version (@ACC(888)) of the instruction or an execution condition that is turned ON only for one scan.



In independent mode, pulse output stops automatically when the specified number of pulses has been output. In continuous mode, pulse output continues until it is stopped from the program.

An error will occur if an attempt is made to switch between independent and continuous mode during pulse output.

PLS2(887) can be executed during pulse output for ACC(888) in either independent or continuous mode, and during acceleration, constant speed, or deceleration. (See note.) ACC(888) can also be executed during pulse output for PLS2(887) during acceleration, constant speed, or deceleration.

Note

(1) Executing PLS2(887) during speed control with ACC(888) (continuous mode) with the same target frequency as ACC(888) can be used to achieved interrupt feeding of a fixed distance. Acceleration will not be performed by PLS2(887) for this application, but if the acceleration rate is set to 0, the Error Flag will turn ON and PLS2(887) will not be executed. Always set the acceleration rate to a value other than 0.

(2) The acceleration/deceleration rate can be specified as 1 Hz or higher. There is no upper limit to the acceleration/deceleration time. If the difference between the starting speed and target speed is more than 100 kHz, the acceleration/deceleration rate will be automatically increased.

- If the difference between the starting speed and target speed is between 100 and 200 KHz, the acceleration/deceleration rate will be 2 Hz or higher.
- If the difference between the starting speed and target speed is between 200 and 300 KHz, the acceleration/deceleration rate will be 3 Hz or higher.

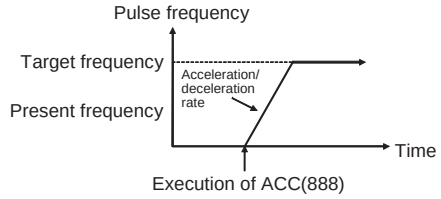
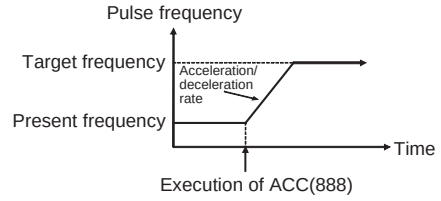
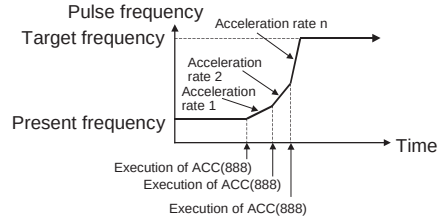
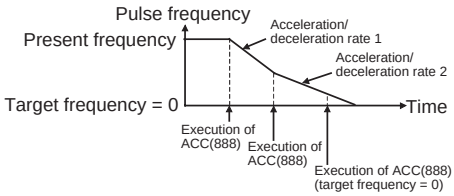
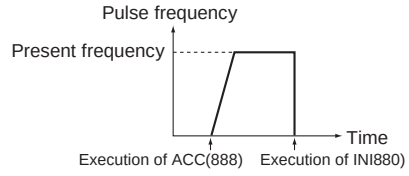
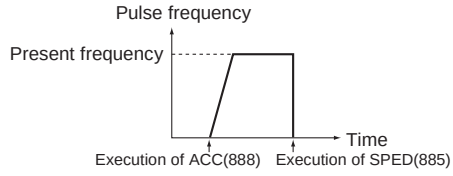
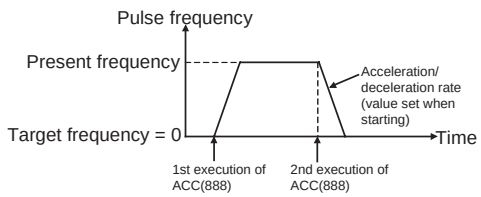
  .
  .
  .

- If the difference between the starting speed and target speed is between 900 and 1,000 KHz, the acceleration/deceleration rate will be 10 Hz or higher.

■ Continuous Mode Speed Control

Pulse output will continue until it is stopped from the program.

Note Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified acceleration and speed	Accelerating the speed (frequency) at a fixed rate		Outputs pulses and changes the frequency at a fixed rate.	ACC(888) (Continuous)
Changing settings	To change speed smoothly	Changing the speed smoothly during operation		Changes the frequency from the present frequency at a fixed rate. The frequency can be accelerated or decelerated.	ACC(888) or SPED(885) (Continuous) ↓ ACC(888) (Continuous)
		Changing the speed in a polyline curve during operation		Changes the acceleration or deceleration rate during acceleration or deceleration.	ACC(888) (Continuous) ↓ ACC(888) (Continuous)
		Stopping while changing the speed		The deceleration rate is changed while stopping. **Note** When the target frequency is changed to 0 Hz, the current deceleration rate will continue to be used.	ACC(888) (Continuous) ↓ ACC(888) (Continuous) ↓ ACC(888) (Continuous, target frequency of 0)
Stopping pulse output	To stop pulse output	Immediate stop		Immediately stops pulse output.	ACC(888) (Continuous) ↓ INI(880) (Continuous)
	To stop pulse output	Immediate stop		Immediately stops pulse output.	ACC(888) (Continuous) ↓ SPED(885) (Continuous, target frequency of 0)
	To stop pulse output smoothly	Decelerating to a stop		Pulse output is decelerated to stop. **Note** If the target frequency of the 2nd execution of ACC(888) is 0 Hz, then the deceleration rate of the 1st ACC(888) will be used.	ACC(888) (Continuous) ↓ ACC(888) (Continuous, target frequency of 0)

■ Independent Mode Positioning

When independent mode operation is started, pulse output will be continued until the specified number of pulses has been output.

The deceleration point is calculated from the number of output pulses and deceleration rate set in S and when that point is reached, the frequency is decreased every pulse control period (4 ms) at the deceleration rate specified in S until the specified number of points has been output, at which point pulse output is stopped.

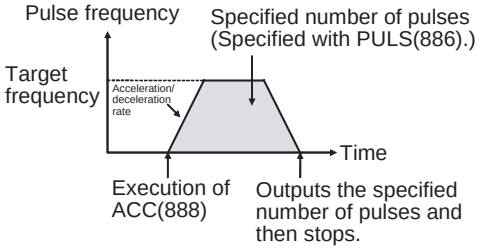
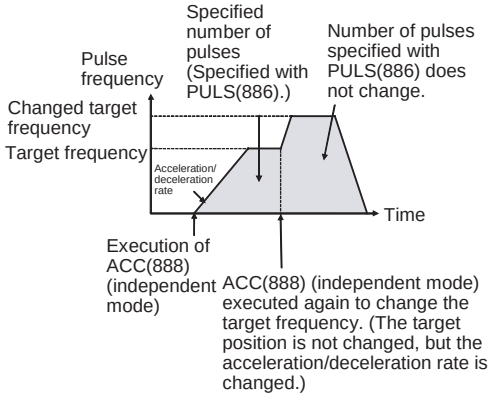
Note

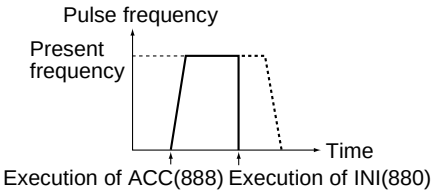
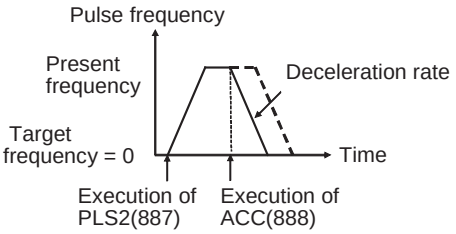
(1) Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

(2) The number of output pulses must be set each time output is restarted.

(3) The number of output pulses must be set in advance with PULS(881). Pulses will not be output for ACC(888) if PULS(881) is not executed first.

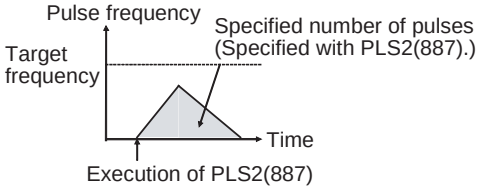
(4) The direction set in the ACC(888) operand will be ignored if the number of pulses is set with PULS(881) as an absolute value.

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	Simple trapezoidal control	Positioning with trapezoidal acceleration and deceleration (Same rate used for acceleration and deceleration; no starting speed) The number of pulses cannot be changed during positioning.		Accelerates and decelerates at the same fixed rate and stops immediately when the specified number of pulses has been output. (See note.) **Note** The target position (specified number of pulses) cannot be changed during positioning.	PULS(886) ↓ ACC(888) (Independent)
Changing settings	To change speed smoothly (with the same acceleration and deceleration rates)	Changing the target speed (frequency) during positioning (acceleration rate = deceleration rate)		ACC(888) can be executed during positioning to change the acceleration/deceleration rate and target frequency. The target position (specified number of pulses) is not changed.	PULS(886) ↓ ACC(888) or SPED(885) (Independent) ↓ ACC(888) (Independent)

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output	To stop pulse output. (Number of pulses setting is not preserved.)	Immediate stop		Pulse output is stopped immediately and the remaining number of output pulses is cleared.	PULS(886) ↓ ACC(888) (Independent) ↓ INI(880)
	To stop pulse output smoothly. (Number of pulses setting is not preserved.)	Decelerating to a stop		Decelerates the pulse output to a stop. **Note** If ACC(888) started the operation, the original acceleration/ deceleration rate will remain in effect. If SPED(885) started the operation, the acceleration/ deceleration rate will be invalid and the pulse output will stop immediately.	PULS(886) ↓ ACC(888) or SPED(885) (Independent) ↓ ACC(888) (Independent, independent, target frequency of 0) PLS2(887) ↓ ACC(888) (Independent, target frequency of 0)

Note Triangular Control

If the specified number of pulses is less than the number required to reach the target frequency and return to zero, the function will automatically reduce the acceleration/deceleration time and perform triangular control (acceleration and deceleration only.) An error will not occur.

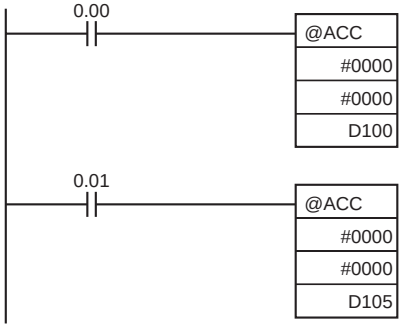


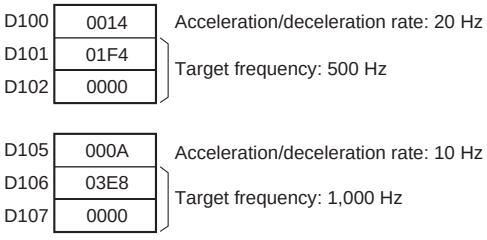
Flags

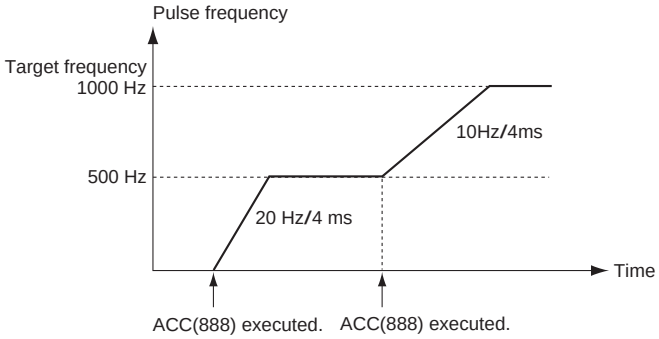
Name	Label	Operation
Error Flag	ER	ON if the specified range for P, M, or S is exceeded. ON if pulses are being output using ORG(889) for the specified port. ON if ACC(888) is executed to switch between independent and continuous mode for a port that is outputting pulses for SPED(885), ACC(888), or PLS2(887). ON if ACC(888) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if ACC(888) is executed for an absolute pulse output in independent mode but the origin has not been established.

Example

When CIO 0.00 turns ON in the following programming example, ACC(888) starts pulse output from pulse output 0 in continuous mode in the clockwise direction using the CW/CCW method. Pulse output is accelerated at a rate of 20 Hz every 4 ms until the target frequency of 500 Hz is reached. When CIO 0.01 turns ON, ACC(888) changes to an acceleration rate of 10 Hz every 4 ms until the target frequency of 1,000 Hz is reached.







3-20-9 ORIGIN SEARCH: ORG(889)

Purpose

ORG(889) performs an origin search or origin return operation.

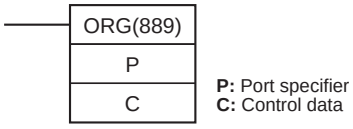
■ Origin Search

Pulses are output using the specified method to actually drive the motor and establish the origin based on origin proximity input and origin input signals.

■ Origin Return

The positioning system is returned to the pre-established origin.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ORG(889)
	Executed Once for Upward Differentiation	@ORG(889)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

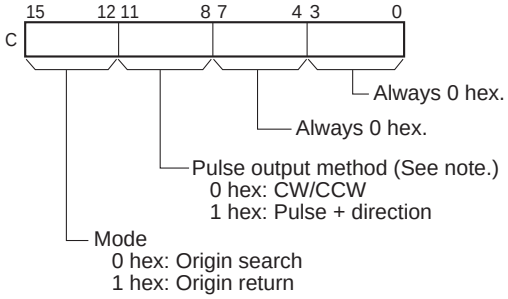
P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2
0003 hex	Pulse output 3

C: Control Data

The value of C determines the origin search method.



Note: Use the same pulse output method when using both pulse outputs 0 and 1.

Operand Specifications

Area	P	C
CIO Area	---	---
Work Area	---	---
Holding Bit Area	---	---
Auxiliary Bit Area	---	---
Timer Area	---	---
Counter Area	---	---
DM Area	---	---
Indirect DM addresses in binary	---	---
Indirect DM addresses in BCD	---	---
Constants	See description of operand.	See description of operand.
Data Registers	---	---
Index Registers	---	---
Indirect addressing using Index Registers	---	---

Description

ORG(889) performs an origin search or origin return operation for the port specified in P using the method specified in C.

The following parameters must be set in the PLC Setup before ORG(889) can be executed. Refer to the *CP1H Operation Manual* for details.

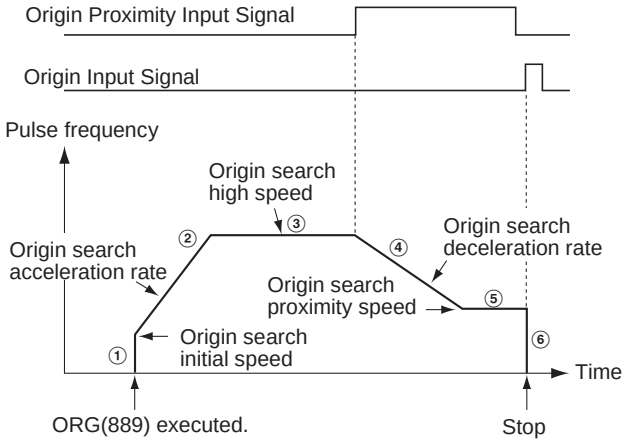
Origin search	Origin return
Origin Search Function Enable/Disable	Origin Search/Return Initial Speed
Origin Search Operating Mode	Origin Return Target Speed
Origin Search Operation Setting	Origin Return Acceleration Rate
Origin Detection Method	Origin Return Deceleration Rate
Origin Search Direction Setting	
Origin Search/Return Initial Speed	
Origin Search High Speed	
Origin Search Proximity Speed	
Origin Compensation	
Origin Search Acceleration Rate	
Origin Search Deceleration Rate	
Limit Input Signal Type	
Origin Proximity Input Signal Type	
Origin Input Signal Type	

An origin search or origin return is started each time ORG(889) is executed. It is thus normally sufficient to use the differentiated version (@ORG(889)) of the instruction or an execution condition that is turned ON only for one scan.

■ Origin Search (Bits 12 to 15 of C = 0 hex)

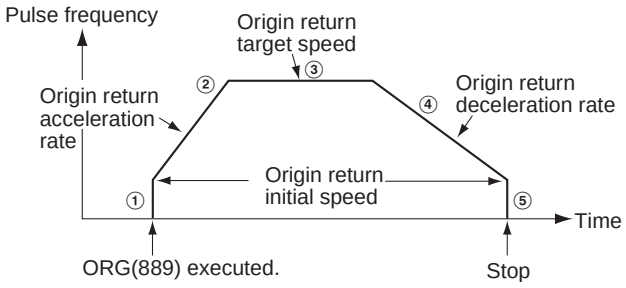
ORG(889) starts outputting pulses using the specified method at the Origin Search Initial Speed (1 in diagram). Pulse output is accelerated to the Origin Search High Speed using the Origin Search Acceleration Rate (2 in diagram). Pulse output is then continued at constant speed until the Origin Proximity Input Signal turns ON (3 in diagram), from which point pulse output is decelerated to the Origin Search Proximity Speed using the Origin Search Deceleration Rate (4 in diagram). Pulses are then output at constant speed until the Origin Input Signal turns ON (5 in diagram). Pulse output is stopped when the Origin Input Signal turns ON (6 in diagram).

When the origin search operation has been completed, the Error Counter Reset Output will be turned ON. The above operation, however, depends on the operating mode, origin detection method, and other parameters. Refer to the *CP1H Operation Manual* for details.



■ Origin Return (Bits 12 to 15 of C = 1 hex)

ORG(889) starts outputting pulses using the specified method at the Origin Return Initial Speed (1 in diagram). Pulse output is accelerated to the Origin Return Target Speed using the Origin Return Acceleration Rate (2 in diagram) and pulse output is continued at constant speed (3 in diagram). The deceleration point is calculated from the number of pulses remaining to the origin and the deceleration rate and when that point is reached, the pulse output is decelerated (4 in diagram) at the Origin Return Deceleration Rate until the Origin Return Start Speed is reached, at which point pulse output is stopped at the origin (5 in diagram).

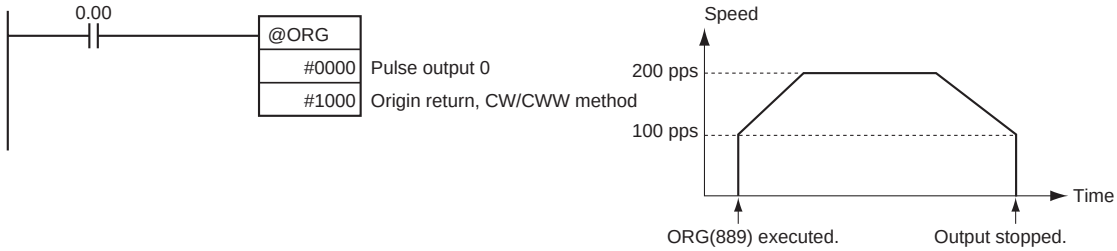


Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P or C is exceeded. ON if ORG(889) is specified for a port during pulse output for SPED(885), ACC(888), or PLS2(887). ON if ORG(889) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if the origin search or origin return parameters set in the PLC Setup are not within range. ON if the Origin Search High Speed is less than or equal to the Origin Search Proximity Speed or the Origin Search Proximity Speed is less than or equal to the Origin Search Initial Speed. ON if the Origin Return Target speed is less than or equal to the Origin Return Initial Speed. ON if an origin return operation is attempted when the origin has not been established.

Example

When CIO 0.00 turns ON in the following programming example, ORG(889) starts an origin return operation for pulse output 0 by outputting pulses using the CW/CCW method. According to the PLC Setup, the initial speed is 100 pps, the target speed is 200 pps, and the acceleration and deceleration rates are 50 Hz/4 ms.



The PLC Setup parameters are as follows:

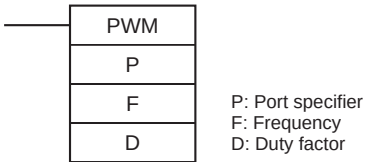
Parameter	Setting
Pulse Output 0 Starting Speed for Origin Search and Origin Return	0000 0064 hex: 100 pps
Pulse Output 0 Origin Return Target Speed	0000 00C8 hex: 200 pps
Pulse Output 0 Origin Return Acceleration Rate	0032 hex: 50 hex/4 ms
Pulse Output 0 Origin Return Deceleration Rate	0032 hex: 50 hex/4 ms

3-20-10 PULSE WITH VARIABLE DUTY FACTOR: PWM(891)

Purpose

PWM(891) is used to output pulses with the specified duty factor from the specified port.

Ladder Symbol



Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	PWM(891)
	Executed Once for Upward Differentiation	@PWM(891)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		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
1000 hex	Pulse output 0 (duty factor: in increments of 0.1%)
1001hex	Pulse output 1 (duty factor: in increments of 0.1%)

F: Frequency

F specifies the frequency of the pulse output between 0.1 and 6,553.5 Hz (0.1 Hz units, 0001 to FFFF hex). The accuracy of the PMW(891) waveform that is actually output (ON duty +5%/–0%) applies only to 0.1 to 1,000.0 Hz due to limitations in the output circuits.

D: Duty Factor

D specifies the duty factor of the pulse output, i.e., the percentage of time that the output is ON. The value of D must be between the following range.

- 0.0% and 100.0% (0.1% units, 0000 to 03E8 hex)

Operand Specifications

Area	P	F	D
CIO Area	---	CIO 0 to CIO 6143	CIO 0 to CIO 6143
Work Area	---	W0 to W511	W0 to W511
Holding Bit Area	---	H0 to H511	H0 to H511
Auxiliary Bit Area	---	A448 to A959	A448 to A959
Timer Area	---	T0000 to T4095	T0000 to T4095
Counter Area	---	C0000 to C4095	C0000 to C4095

Area	P	F	D
DM Area	---	D0 to D32767	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767	*D0 to *D32767
Constants	See description of operand.	0000 to FFFF hex	0000 to 03E8 hex
Data Registers	---	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

PWM(891) outputs the frequency specified in F at the duty factor specified in D from the port specified in P. PWM(891) can be executed during duty-factor pulse output to change the duty factor without stopping pulse output. Any attempts to change the frequency will be ignored.

Pulse output is started each time PWM(891) is executed. It is thus normally sufficient to use the differentiated version (@PWM(891)) of the instruction or an execution condition that is turned ON only for one scan.

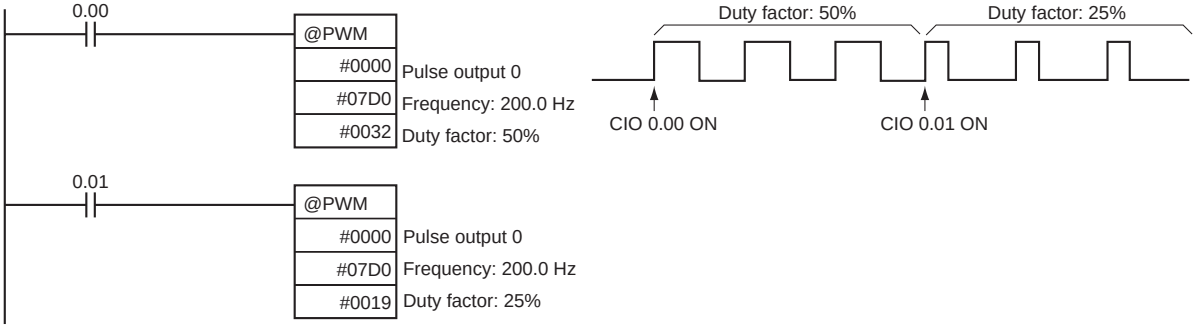
The pulse output will continue either until INI(880) is executed to stop it (C = 0003 hex: stop pulse output) or until the CPU Unit is switched to PROGRAM mode.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, F, or D is exceeded. ON if pulses are being output using ORG(889) for the specified port. ON if PWM(891) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task.

Example

When CIO 0.00 turns ON in the following programming example, PWM(891) starts pulse output from pulse output 0 at 200 Hz with a duty factor of 50%. When CIO 0.01 turns ON, the duty factor is changed to 25%.



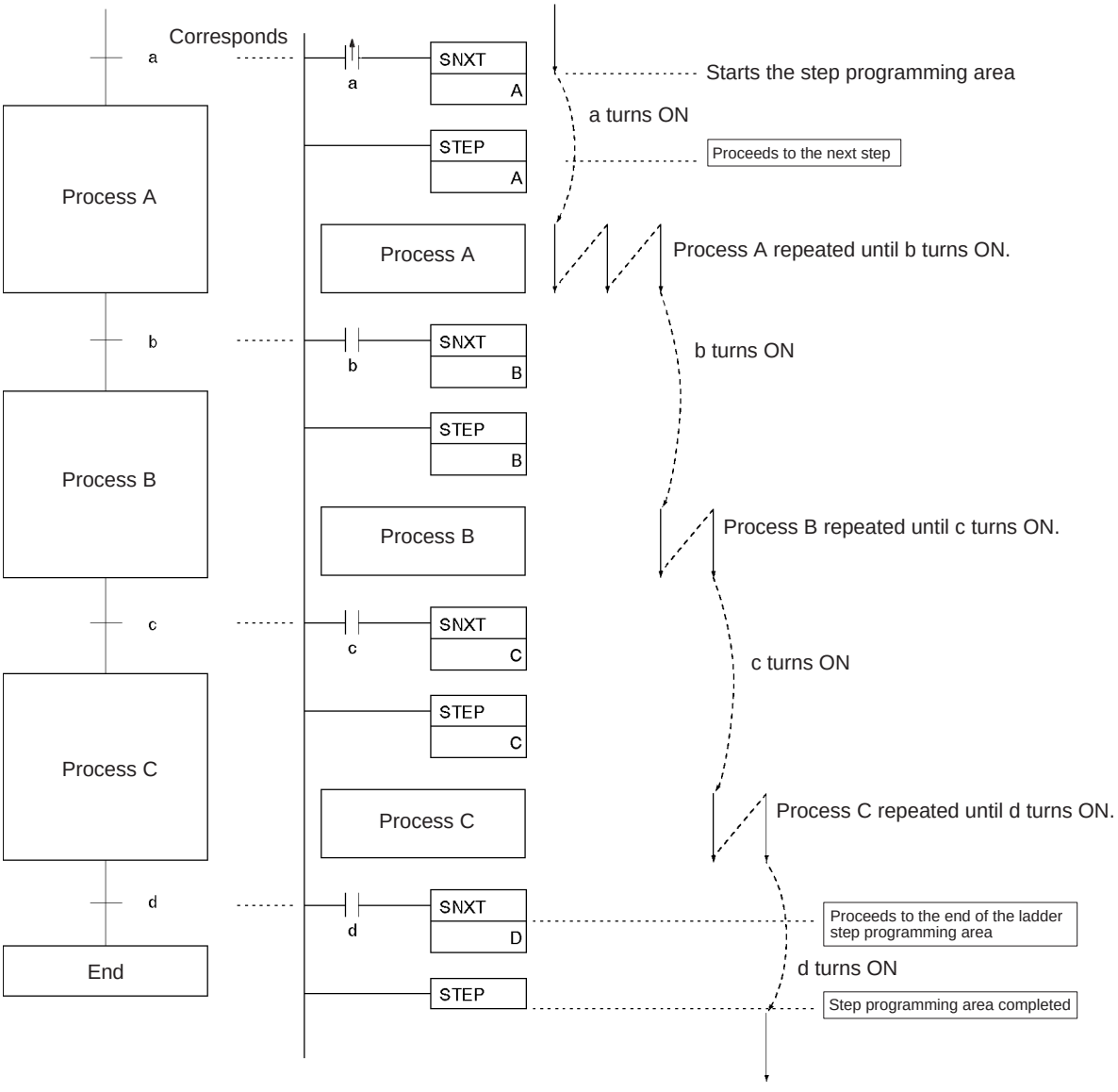
3-21 Step Instructions

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	752
STEP START	SNXT	009	752

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.

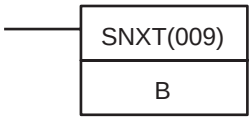
3-21-1 STEP DEFINE and STEP START: STEP(008)/SNXT(009)

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.

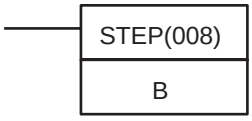
STEP(008) is placed immediately after the SNXT(009) instruction and before each process. It defines the start of each process and specified the control bit for it. It is also placed at the end of the step programming area after the last SNXT(009) to indicate the end of the step programming area. When 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.
Immediate Refreshing Specification		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	W0.00 to W511.15
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Registers	---

Area	B
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

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 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.

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, TIMX(551), TIMH(015), TIMHX(551), TMHH(540), TIMHHX(552), TIML(542), and TIMLX(553)	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 TTIM(087), TTIMX(555), MTIM(543), MTIMX(554), SET, REST, CNT, CNTX(546), CNTR(012), CNTRX(548), 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, as 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)

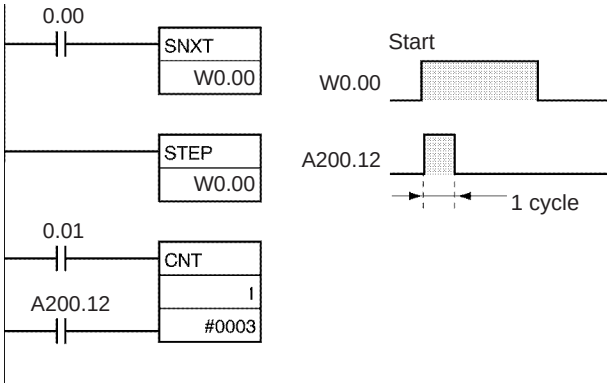
STEP(008) and SNXT(009) cannot be used inside of subroutines, interrupt programs, or block programs.

Be sure that two steps are not executed during the same cycle.

Instructions that Cannot be Used Within Step Programs

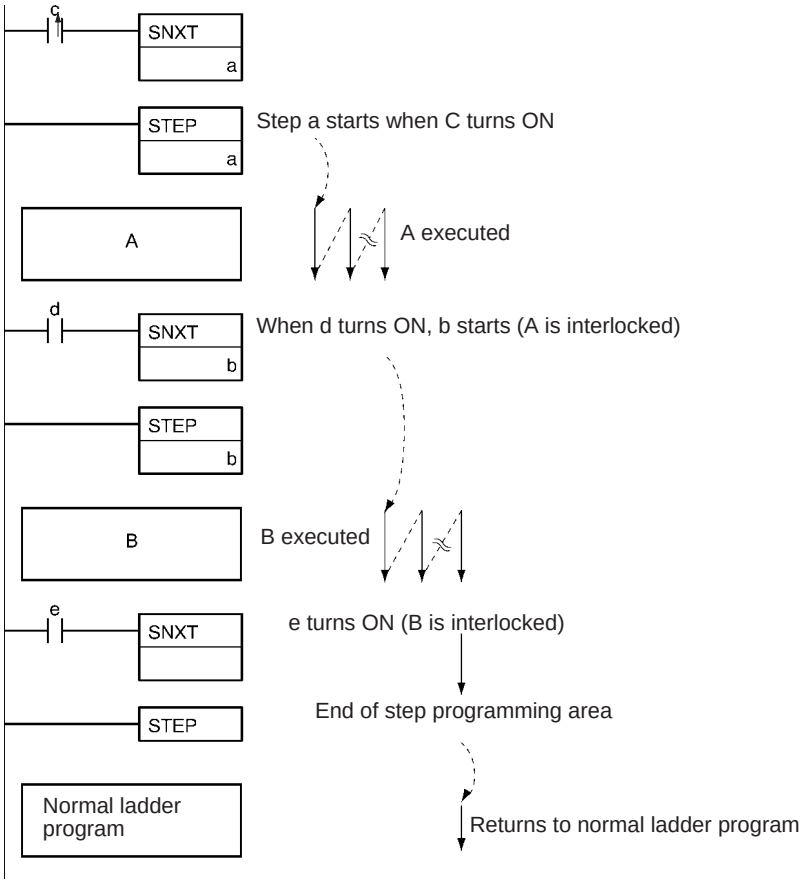
The instructions that cannot be used within step programs are listed in the following table.

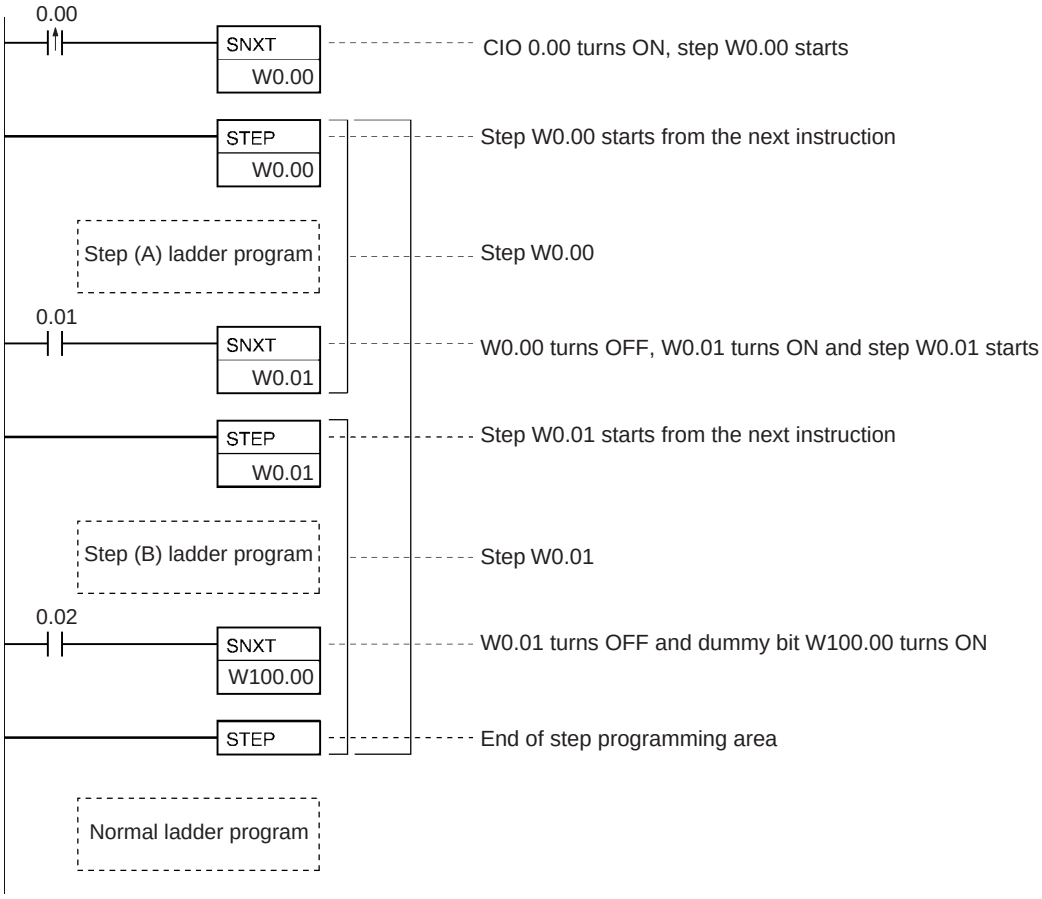
Function	Mnemonic	Name
Sequence Control Instructions	END(001)	END
	IL(002)	INTERLOCK
	ILC(003	INTERLOCK CLEAR
	JMP(004)	JUMP
	JME(005)	JUMP END
	CJP(510)	CONDITIONAL JUMP
	CJPN(511)	CONDITIONAL JUMP NOT
	JMP0(515)	MULTIPLE JUMP
	JME0(516)	MULTIPLE JUMP END
Subroutine Instructions	SBN(092)	SUBROUTINE ENTRY
	RET(093)	SUBROUTINE RETURN



Related Bits

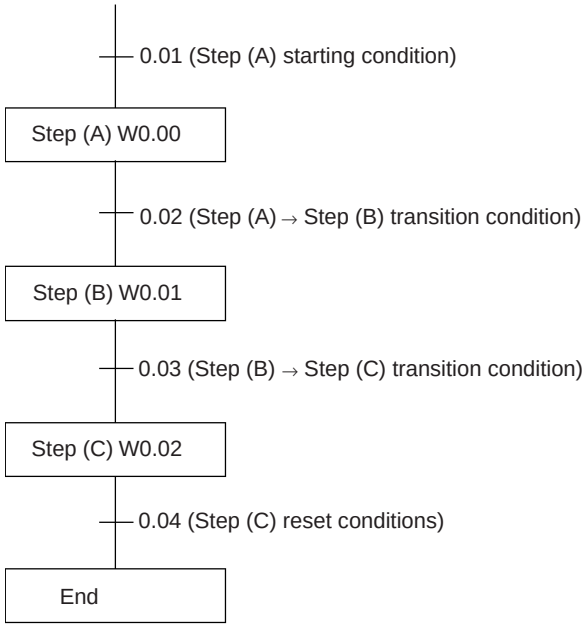
Name	Address	Details
Step Flag	A200.12	ON for one cycle when a step program is started using STEP(008). Can be used to reset timers and perform other processing when starting a new step.

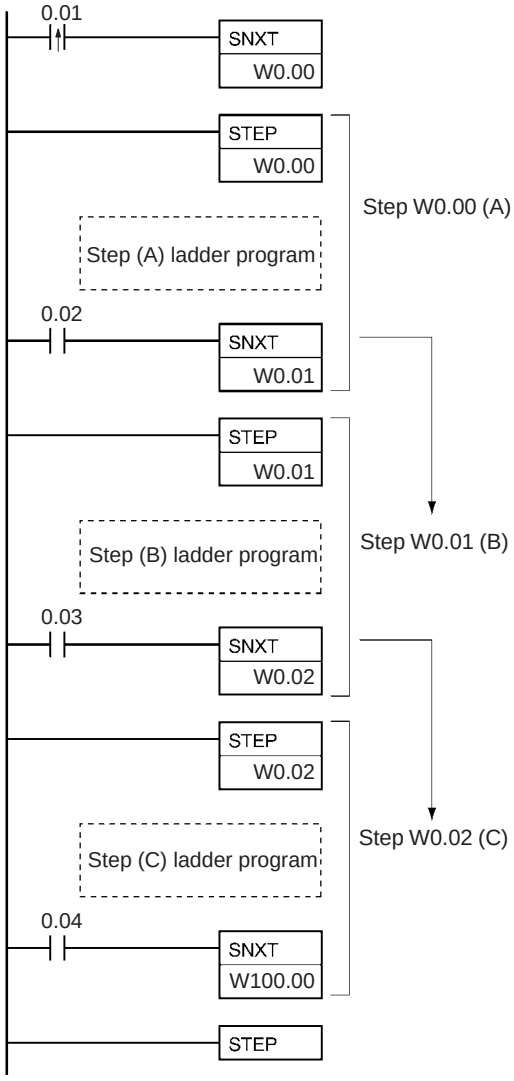




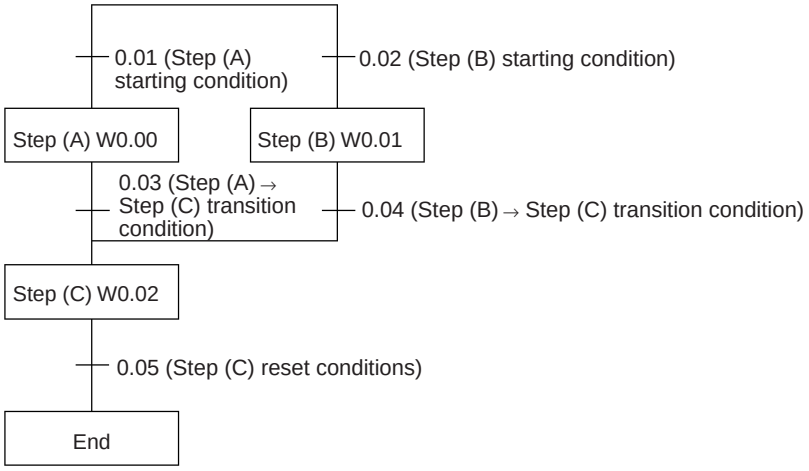
Examples

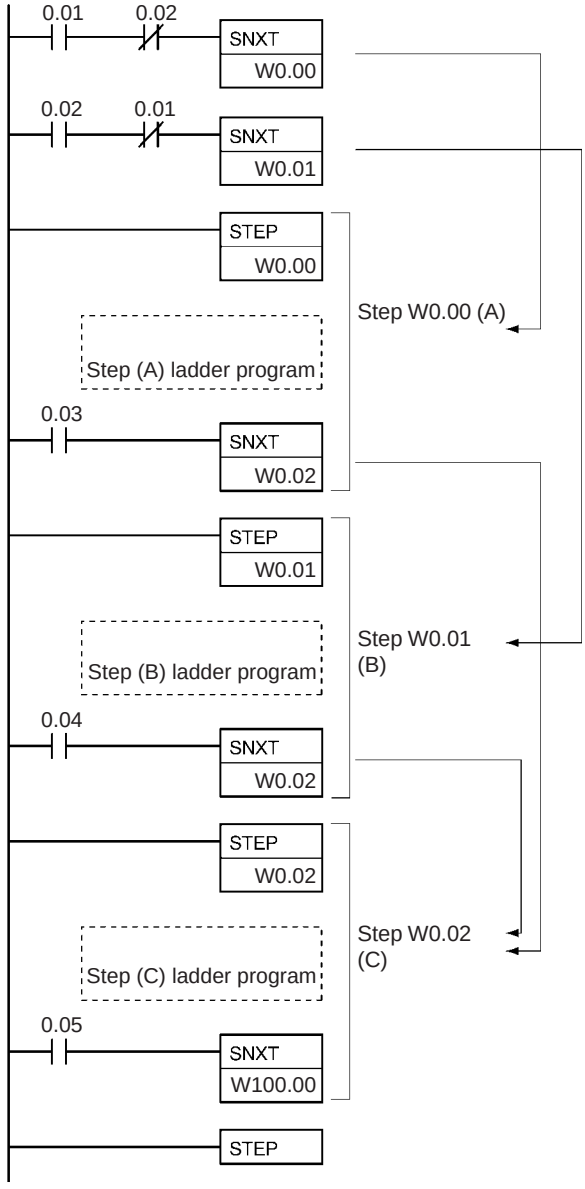
Sequential Control



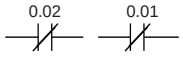


Branching Control



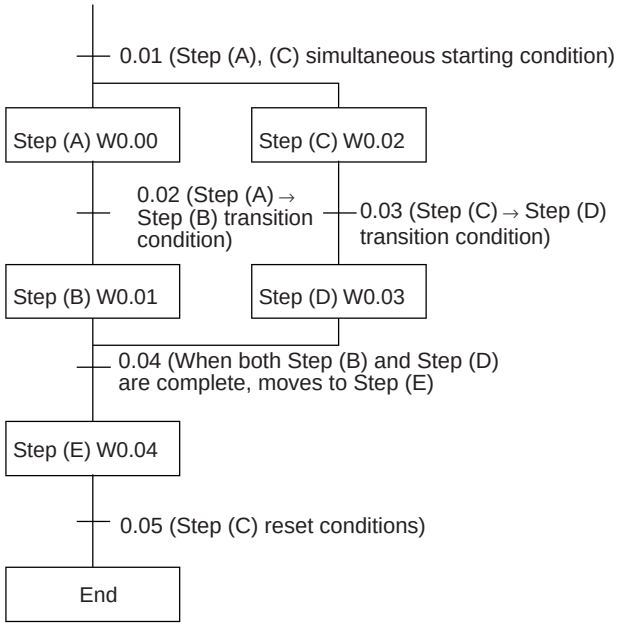


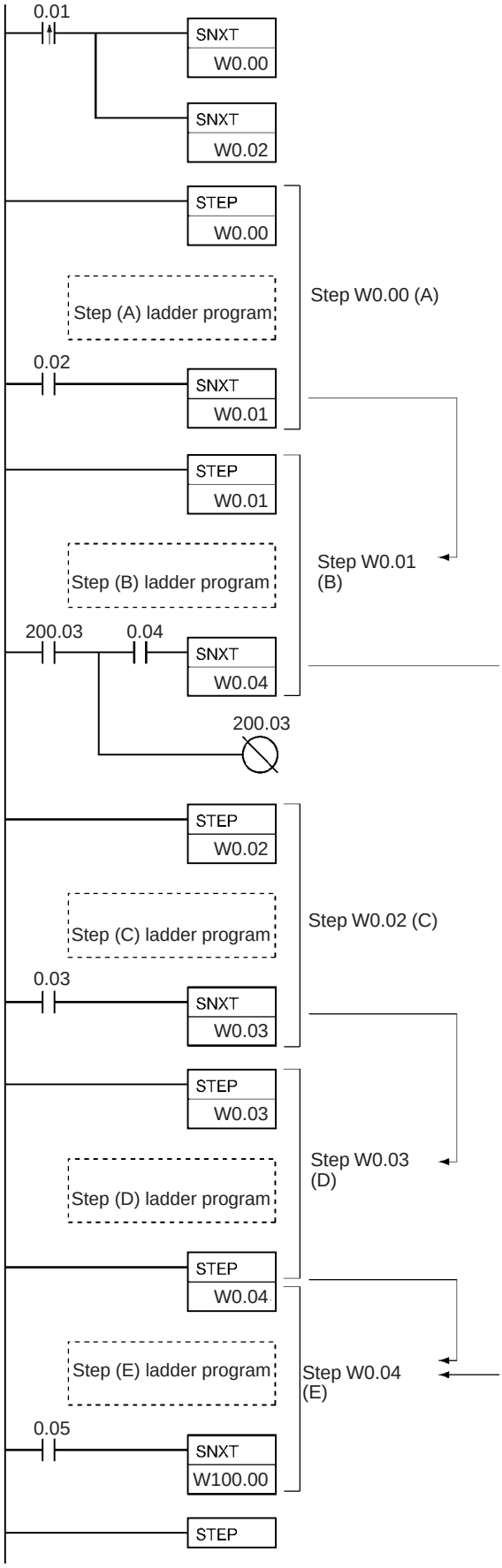
The above programming is used when steps A and B cannot be executed simultaneously. For simultaneous execution of A and B, delete the execution conditions illustrated below.



Note In the above example, where SNXT(009) is executed for W0.02, 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



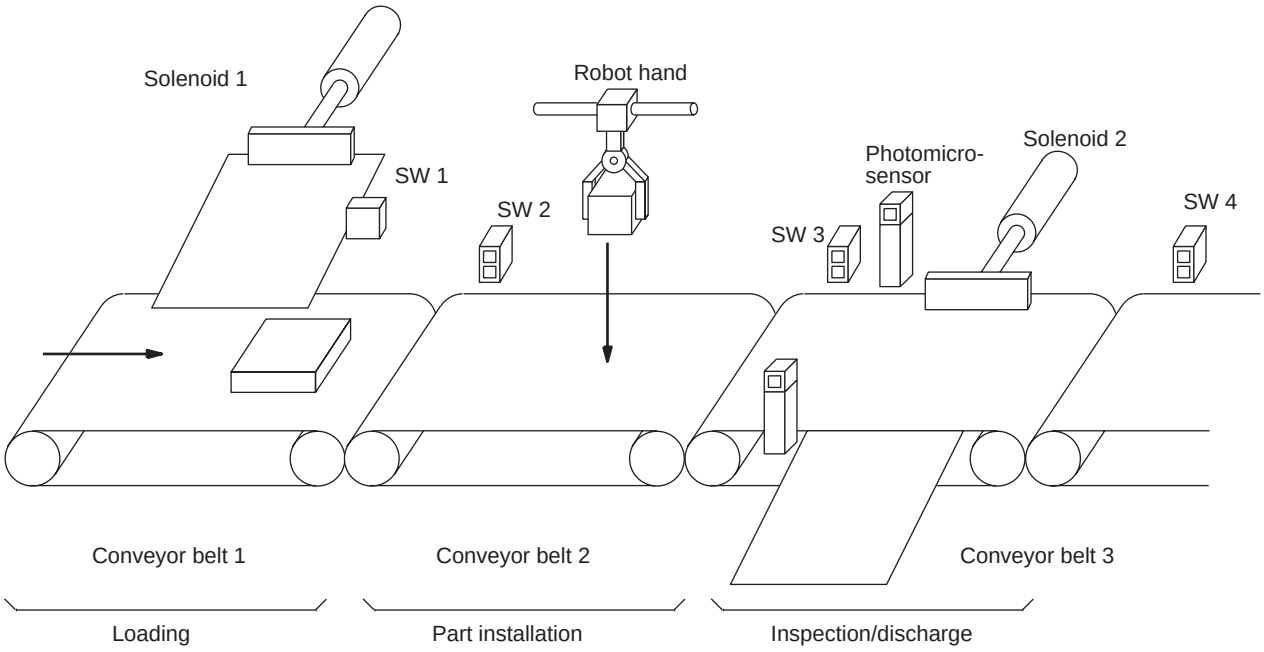

0.01
SNXT
W0.00
SNXT
W0.02
STEP
W0.00
Step (A) ladder program
Step W0.00 (A)
0.02
SNXT
W0.01
STEP
W0.01
Step (B) ladder program
Step W0.01
(B)
200.03
0.04
SNXT
W0.04
200.03
STEP
W0.02
Step (C) ladder program
Step W0.02 (C)
0.03
SNXT
W0.03
STEP
W0.03
Step (D) ladder program
Step W0.03
(D)
STEP
W0.04
Step (E) ladder program
Step W0.04
(E)
0.05
SNXT
W100.00
STEP

Application Examples

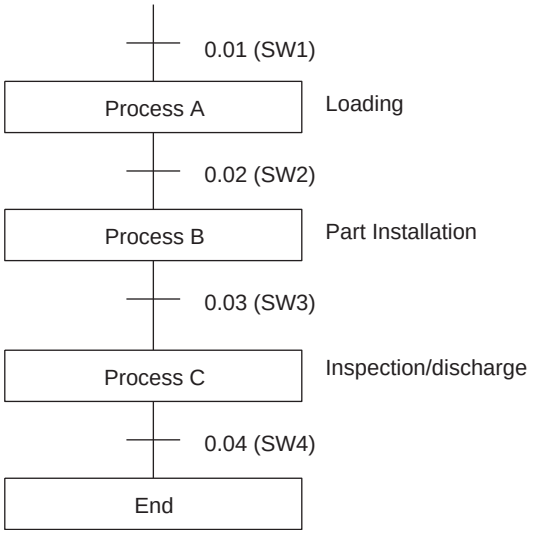
The following three examples demonstrate the three types of execution control possible with step programming. *Example 1* demonstrates sequential execution; *Example 2*, branching execution; and *Example 3*, parallel execution.

Example 1: Sequential Execution

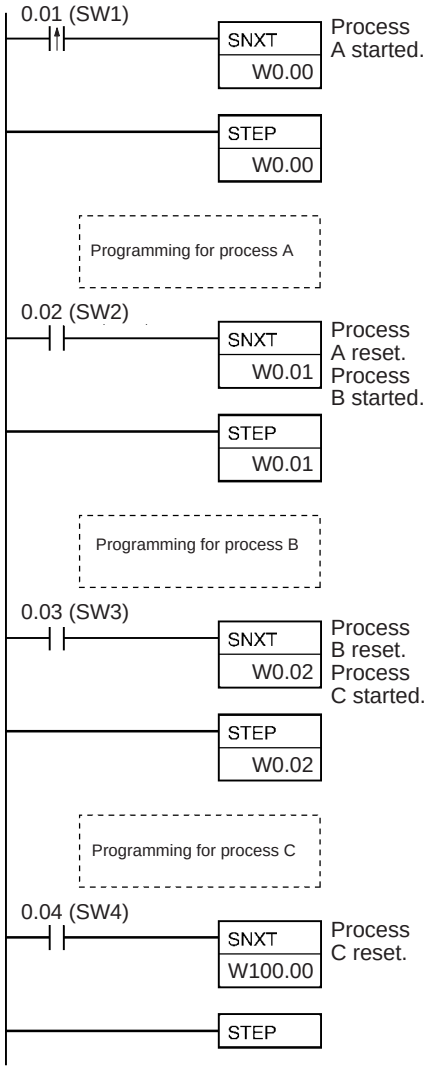
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control.

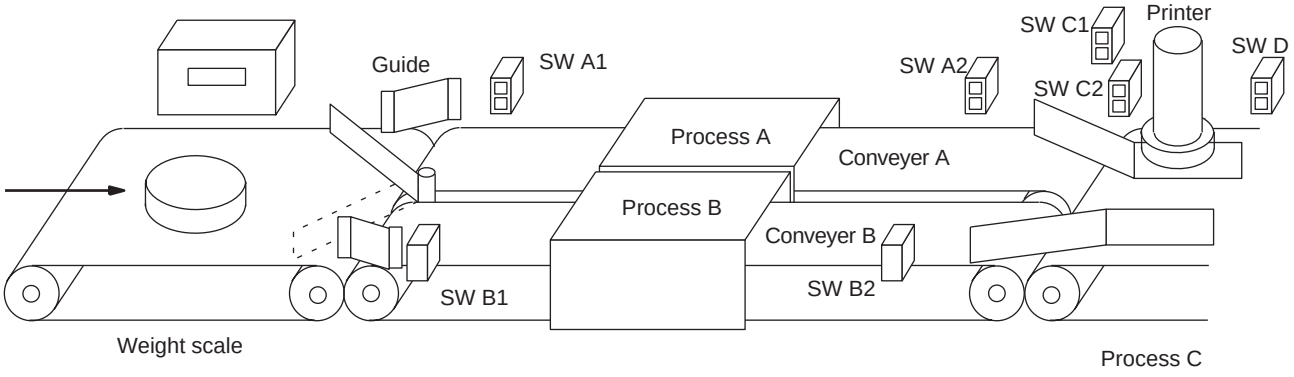


The program for this process, shown below, utilizes the most basic type of step programming: each step is completed by a unique SNXT(009) that starts the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.

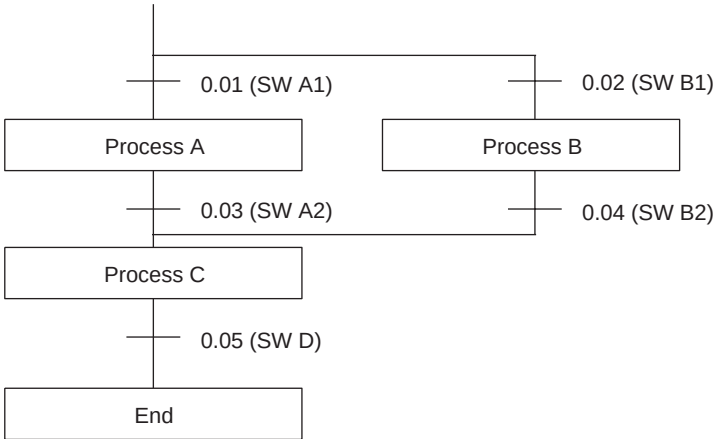


Example 2: Branching Execution

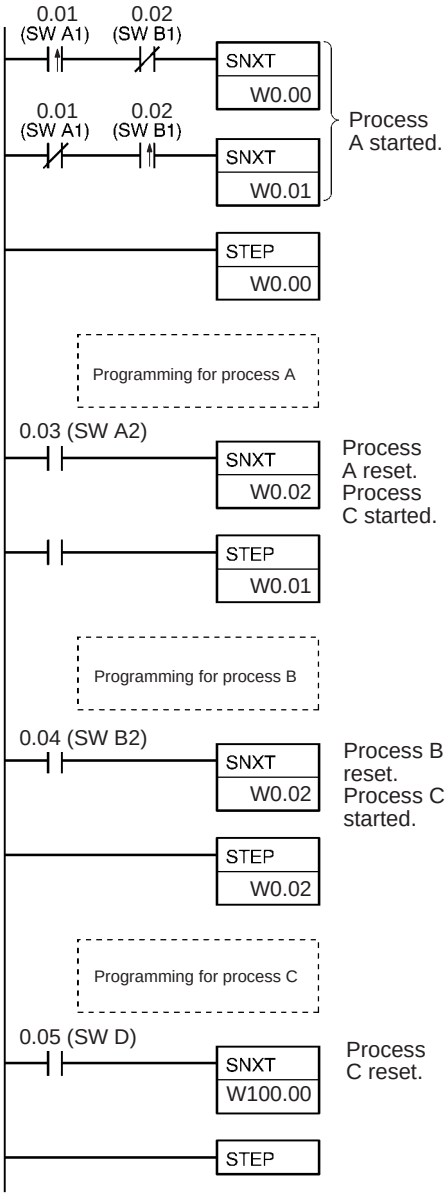
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



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 status of SW A1 and SW B1.

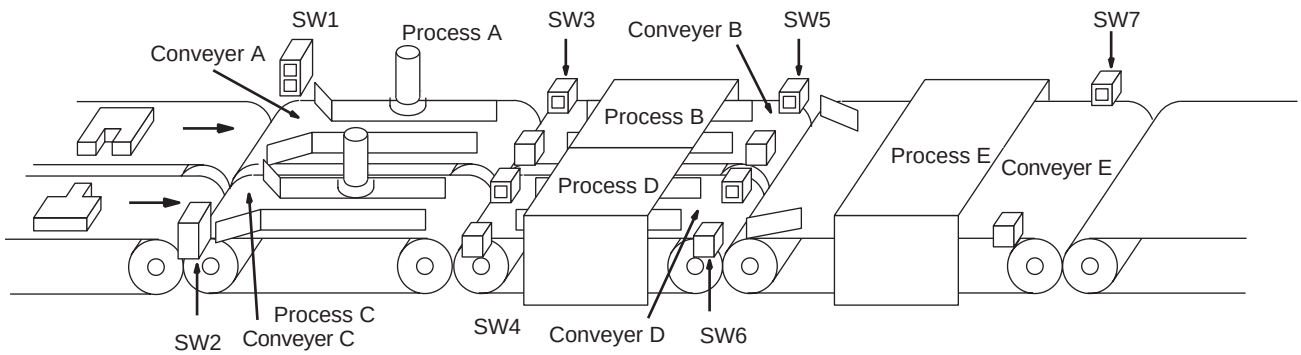


The program for this process, shown below, starts with two SNXT(009) instructions that start processes A and B. Because of the way CIO 0.01 (SW A1) and CIO 0.02 (SW B1) are programmed, only one of these will be executed with an ON execution condition to start either process A or process B. Both of the steps for these processes end with a SNXT(009) that starts the step (process C).

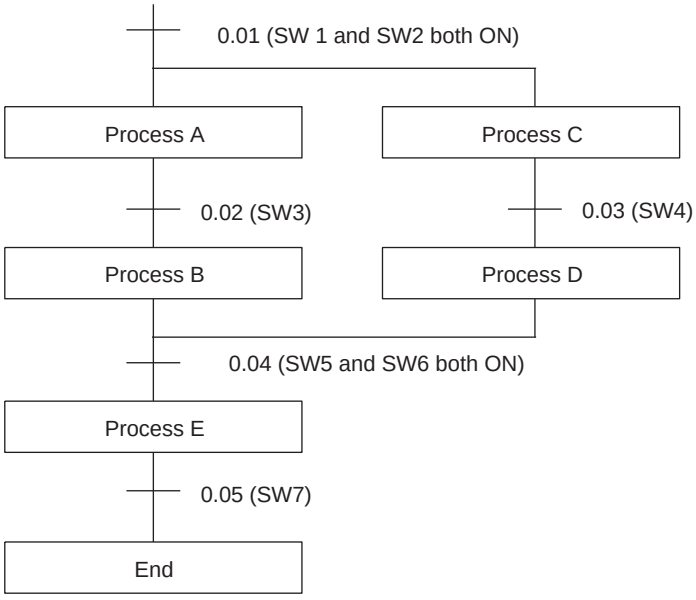


Example 3: Parallel Execution

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.

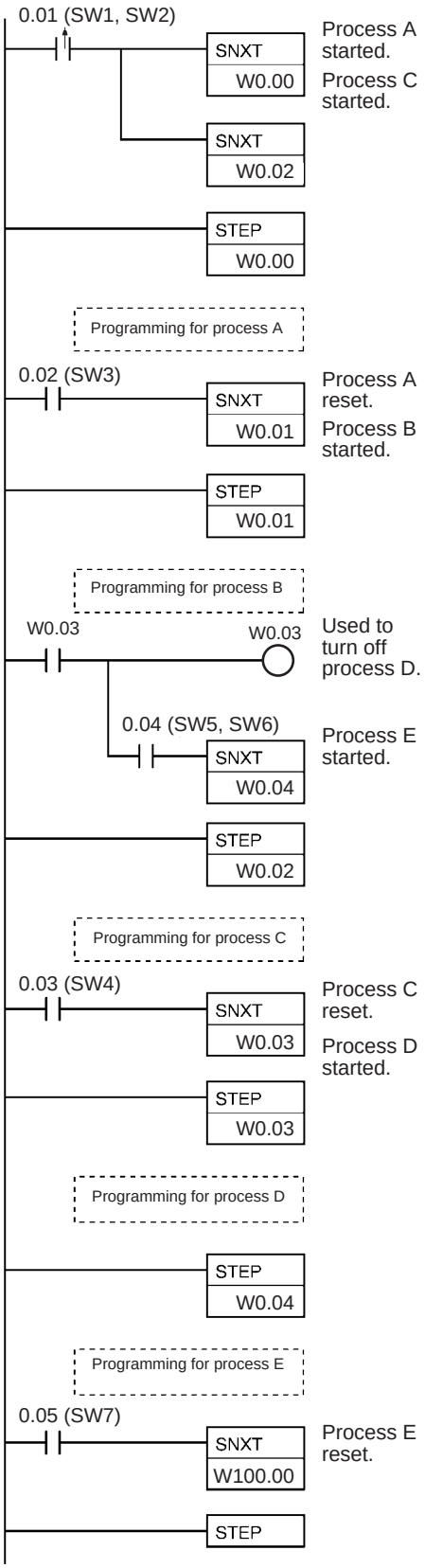


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process 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) W0.04. This is because the OUT for bit W0.03 is in the step reset by SNXT(009) W0.04, i.e., W0.03 is turned OFF when SNXT(009) W0.04 is executed. Process B is thus reset directly and process D is reset indirectly before executing the step for process E.


0.01 (SW1, SW2)
SNXT
W0.00
Process A started.
Process C started.
SNXT
W0.02
STEP
W0.00
Programming for process A
0.02 (SW3)
SNXT
W0.01
Process A reset.
Process B started.
STEP
W0.01
Programming for process B
W0.03
W0.03
Used to turn off process D.
0.04 (SW5, SW6)
SNXT
W0.04
Process E started.
STEP
W0.02
Programming for process C
0.03 (SW4)
SNXT
W0.03
Process C reset.
Process D started.
STEP
W0.03
Programming for process D
STEP
W0.04
Programming for process E
0.05 (SW7)
SNXT
W100.00
Process E reset.
STEP

3-22 Basic I/O Unit Instructions

This section describes instructions used with I/O Units.

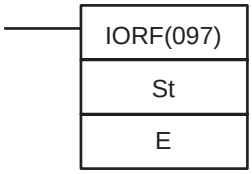
Instruction	Mnemonic	Function code	Page
I/O REFRESH	IORF	097	768
7-SEGMENT DECODER	SDEC	078	771
INTELLIGENT I/O READ	IORD	222	793
INTELLIGENT I/O WRITE	IOWR	223	796
DIGITAL SWITCH INPUT	DSW	210	774
TEN KEY INPUT	TKY	211	778
HEXADECIMAL KEY INPUT	HKY	212	781
MATRIX INPUT	MTR	213	785
7-SEGMENT DISPLAY OUTPUT	7SEG	214	789

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.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

St: Starting Word

CIO 0 to CIO 999 (I/O Bit Area) or
CIO 2000 to CIO 2959 (Special I/O Unit Bit Area)

E: End Word

CIO 0 to CIO 999 (I/O Bit Area) or
CIO 2000 to CIO 2959 (Special I/O Unit Bit Area)

Note St and E must be in the same memory area.

Operand Specifications

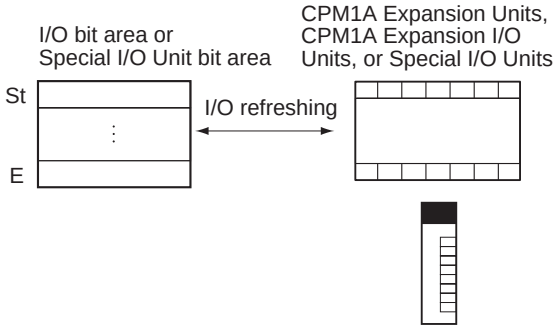
Area	St	E
CIO Area	CIO 0 to CIO 999 CIO 2000 to CIO 2959	
Auxiliary Area	---	
Holding Bit Area	---	
Special Bit Area	---	
Timer Area	---	
Counter Area	---	
DM Area	---	

Area	St	E
Indirect DM addresses in binary	---	
Indirect DM addresses in BCD	---	
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to ,–(– –)IR15	

Description

IORF(097) refreshes the I/O words between St and E, inclusively. IORF(097) is used to refresh words allocated to CPM1A Expansion Units, CPM1A Expansion I/O, or Special I/O Units.

When refreshing is specified for words in the Special I/O Unit bit area, all 10 words allocated to the Unit will be refreshed as long as the first word of the 10 words allocated to the Unit is included in the specified range of words.



IORF(097) cannot be used for built-in I/O on the CPU Unit or for CPU Bus Units. Use instructions with the immediate refresh option. IORF(097) and immediate refresh options cannot be used for built-in analog I/O on XA CPU Units.

If words for which there is no Unit mounted exist between St and E, nothing will be done for those words and only the words allocated to Units will be refreshed.

Applicable Units

The following Units can be refreshed with IORF(097).

I/O	Words	Applicability of IORF(097)
Built-in I/O	Inputs: CIO 0 and CIO 1 Outputs: CIO 100 and CIO 101	No
Built-in analog inputs	CIO 200 to CIO 203	No
Built-in analog outputs	CIO 210 and CIO 211	No
CPM1A Expansion I/O Units	Inputs: CIO 2 to CIO 99 Output: CIO 102 to CIO 199	Yes
CPM1A Expansion Units		
Special I/O Units	CIO 2000 to CIO 2959	Yes
CPU Bus Units	CIO 1500 to CIO 1899	No

Note The Units that can be refreshed with IORF(097) are not necessarily the same as the Units that can be refreshed with immediate refreshing specifications (!).

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.

Precautions

An error will occur if words in both the I/O Bit Area (CIO 0 to CIO 999) and the Special I/O Unit Bit Area (CIO 2000 to CIO 2959) are specified for the same instruction.

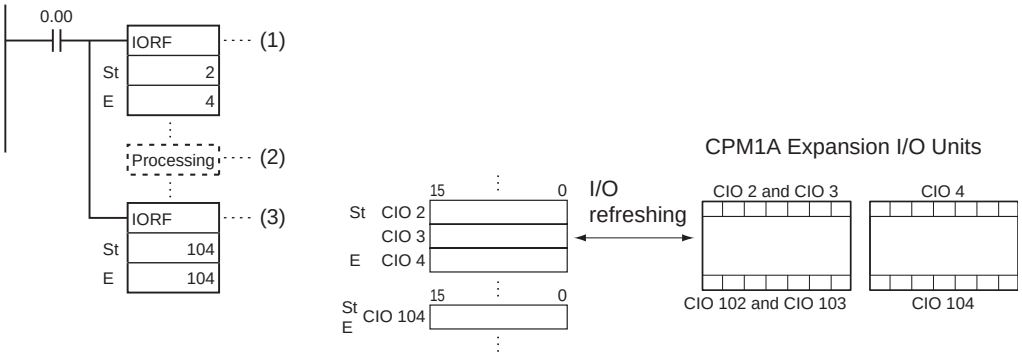
The I/O refreshing initiated by IORF(097) will be stopped midway if an I/O bus error occurs during I/O refreshing.

When IORF(097) is used in an interrupt task, be sure to disable Special I/O Unit cyclic refreshing in the PLC Setup. If cyclic refreshing for Special I/O Units is enabled and I/O refreshing is executed again by IORF(097), a non-fatal Duplicate Refreshing Error will occur and the Interrupt Task Error Flag (A402.13) will be turned ON.

Examples

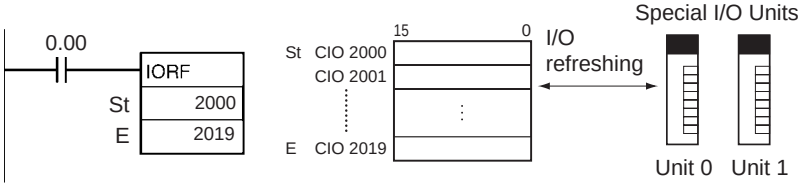
Refreshing Words in the I/O Bit Area

When CIO 0.00 turns ON in the following example, CIO 2 to CIO 4 (36 inputs) are refreshed (1) and then after the required processing is performed (2), CIO 104 (8 outputs) is refreshed.



Refreshing Words in the Special I/O Unit Bit Area

The following example shows how to refresh 20 words from CIO 2000 to CIO 2019 (I/O for Special I/O Units with unit numbers 0 to 1) when CIO 0.00 turns ON.

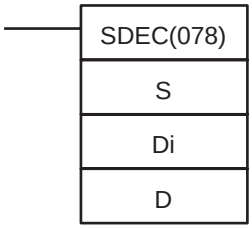


3-22-2 7-SEGMENT DECODER: SDEC(078)

Purpose

Converts the hexadecimal contents of the designated digit(s) into 8-bit, 7-segment display code and places it into the upper or lower 8-bits of the specified destination words.

Ladder Symbol



S: Source word

Di: Digit designator

D: First destination word

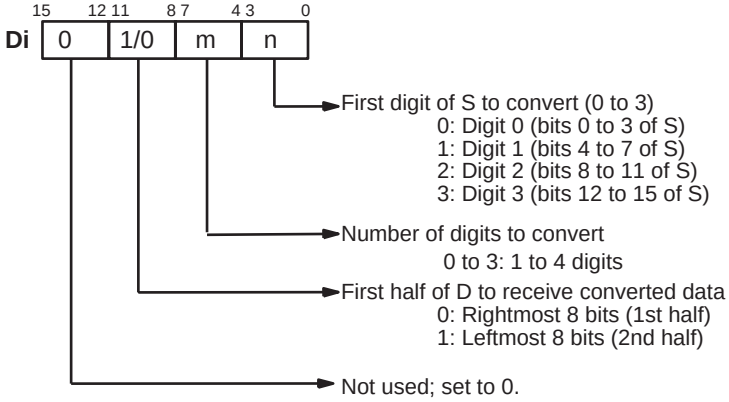
Variations

Variations		
	Executed Each Cycle for ON Condition	SDEC(078)
	Executed Once for Upward Differentiation	@SDEC(078)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands: Digit Designator



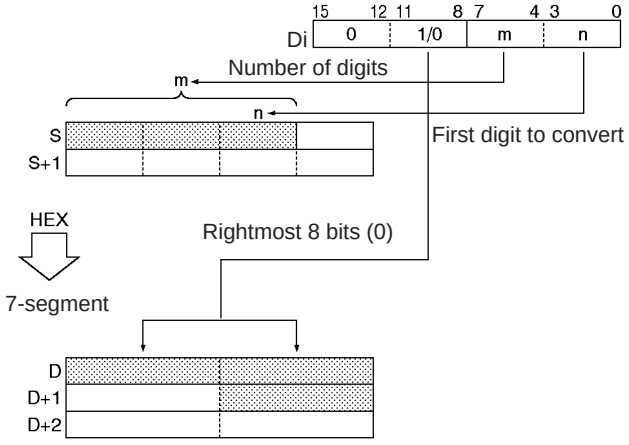
Operand Specifications

Area	S	Di	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@D0 to @D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	---
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SDEC(078) regards the data specified by S as 4-digit hexadecimal data, converts the digits specified in S by Di (first digit and number of digits) to 7-segment data and outputs the results to D in the bits specified in Di.



Flags

Name	Label	Operation
Error Flag	ER	ON if settings in Di are not within the specified ranges. OFF in all other cases.

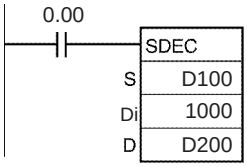
Precautions

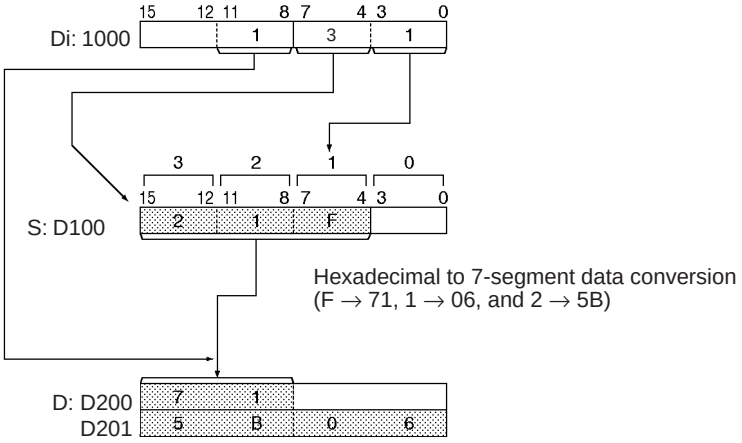
If more than one digit is specified for conversion in Di, digits are converted in order toward the most-significant digit. Digit 0 is the next digit after digit 3.

Results are stored in D in order from the specified portion toward higher-address words. If just one of the bytes in a destination word receives converted data, the other byte is left unchanged.

Examples

When CIO 0.00 turns ON in the following example, the contents of the 3 digits beginning with digit 1 in D100 will be converted from hexadecimal data to 7-segment data, and the results will be output to the upper byte of D200 and both bytes of D201. The specifications of the bytes to be converted and the location of the output bytes are made in CIO 1000.

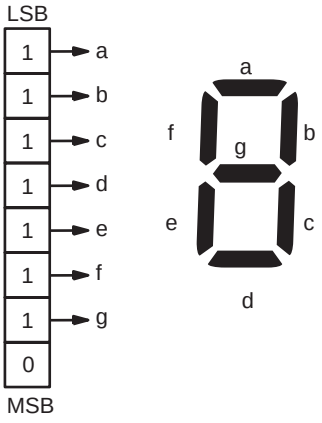




7-segment Data

The following table shows the data conversions from a hexadecimal digit (4 bits) to 7-segment code (8 bits).

Original data					Converted code (segments)									Display Original data
Digit	Bits				–	g	f	e	d	c	b	a	Hex	
0	0	0	0	0	0	0	1	1	1	1	1	1	3F	0
1	0	0	0	1	0	0	0	0	0	1	1	0	06	1
2	0	0	1	0	0	1	0	1	1	0	1	1	5B	2
3	0	0	1	1	0	1	0	0	1	1	1	1	4F	3
4	0	1	0	0	0	1	1	0	0	1	1	0	66	4
5	0	1	0	1	0	1	1	0	1	1	0	1	6D	5
6	0	1	1	0	0	1	1	1	1	1	0	1	7D	6
7	0	1	1	1	0	0	1	0	0	1	1	1	27	7
8	1	0	0	0	0	1	1	1	1	1	1	1	7F	8
9	1	0	0	1	0	1	1	0	1	1	1	1	6F	9
A	1	0	1	0	0	1	1	1	0	1	1	1	77	A
B	1	0	1	1	0	1	1	1	1	1	0	0	7C	b
C	1	1	0	0	0	0	1	1	1	0	0	1	39	C
D	1	1	0	1	0	1	0	1	1	1	1	0	5E	d
E	1	1	1	0	0	1	1	1	1	0	0	1	79	E
F	1	1	1	1	0	1	1	1	0	0	0	1	71	F

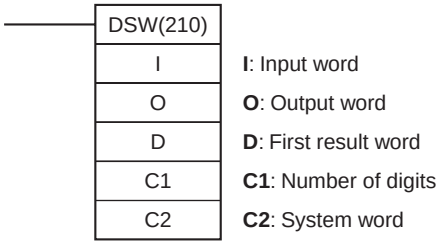


3-22-3 DIGITAL SWITCH INPUT – DSW(210)

Purpose

Reads the value set on a external digital switch (or thumbwheel switch) connected to an I/O Unit and stores the 4-digit or 8-digit value in the specified words.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	DSW(210)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

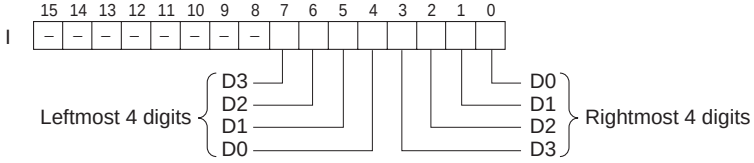
Applicable Program Areas

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

Operands

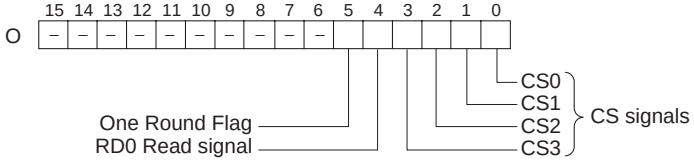
I: Input Word (Data Line D0 to D3 Inputs)

Specify the input word allocated to the Input Unit and connect the digital switch's D0 to D3 data lines to the Input Unit as shown in the following diagram.



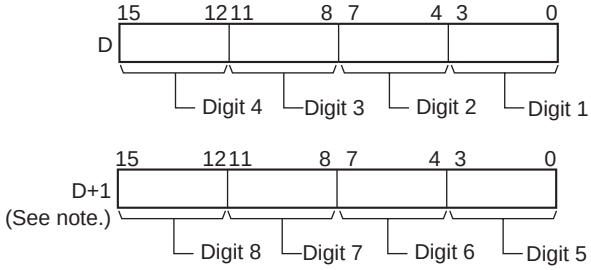
O: Output Word (CS/RD Control Signal Outputs)

Specify the output word allocated to the Output Unit and connect the digital switch's control signals (CS and RD signals) to the Output Unit as shown in the following diagram.



D: First Result Word

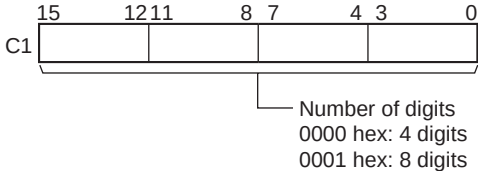
Specifies the leading word address where the external digital switch's set values will be stored.



Note: Only when C1 = 0001 hex to read 8 digits.

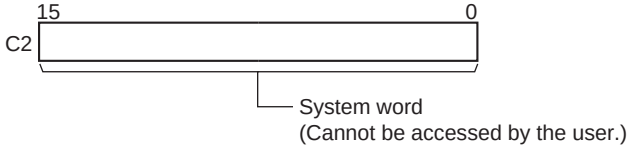
C1: Number of Digits

Specifies the number of digits that will be read from the external digital switch. Set C1 to 0000 hex to read 4 digits or 0001 hex to read 8 digits.



C2: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	I	O	D	C1	C2
CIO Area	CIO 0 to CIO 6143			---	CIO 0 to CIO 6143
Work Area	W0 to W511			---	W0 to W511
Holding Bit Area	H0 to H511			---	H0 to H511
Auxiliary Bit Area	A0 to A959	A448 to A953		---	A448 to A959
Timer Area	T0000 to T4095			---	T0000 to T4095
Counter Area	C0000 to C4095			---	C0000 to C4095
DM Area	D0 to D32767			---	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767			---	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767			---	---
Constants	---			0000 or 0001 hex	---
Data Registers	DR0 to DR15		---	---	DR0 to DR15

Area	I	O	D	C1	C2
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		---	---	,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

DSW(210) outputs control signals to bits 00 to 04 of O, reads the specified number of digits (either 4-digit or 8-digit, specified in C1) of digital switch data line data from I, and stores the result in D and D+1. (If 4 digits are read, the result is stored in D. If 8 digits are read, the result is stored in D and D+1.)

DSW(210) reads the 4-digit or 8-digit switch data once every 16 cycles, and then starts over and continues reading the data. The One Round Flag (bit 05 of O) is turned ON once every 16 CPU Unit cycles.

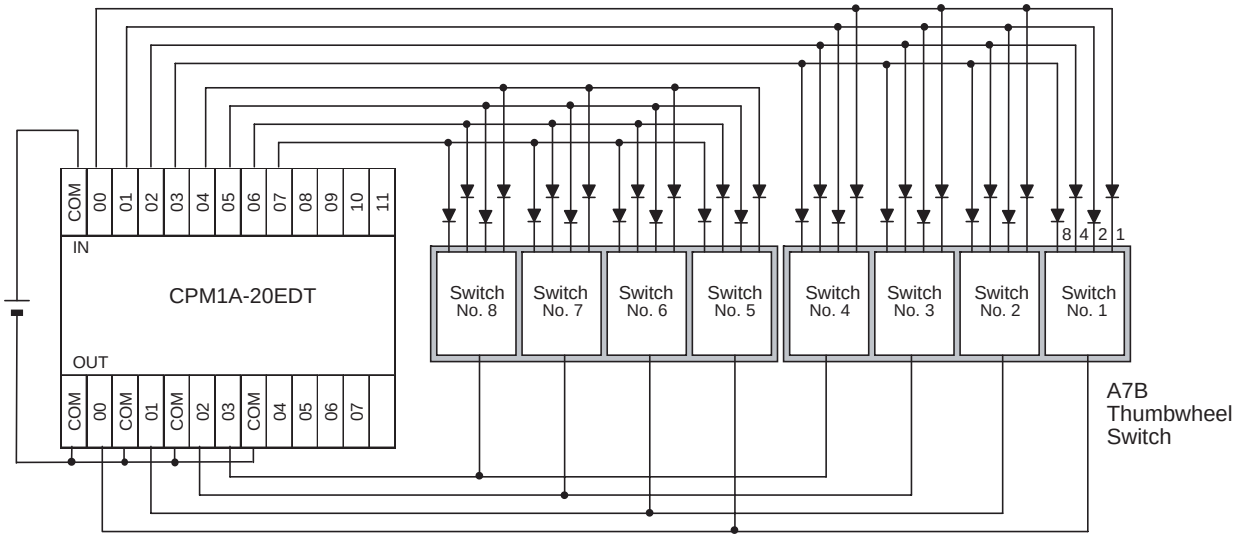
DSW(210) reads the 4-digit or 8-digit data once in 16 cycles, and then starts over and reads the data again in the next 16 cycles.

When executed, DSW(210) begins reading the switch data from the first of the sixteen cycles, regardless of the point at which the last instruction was stopped.

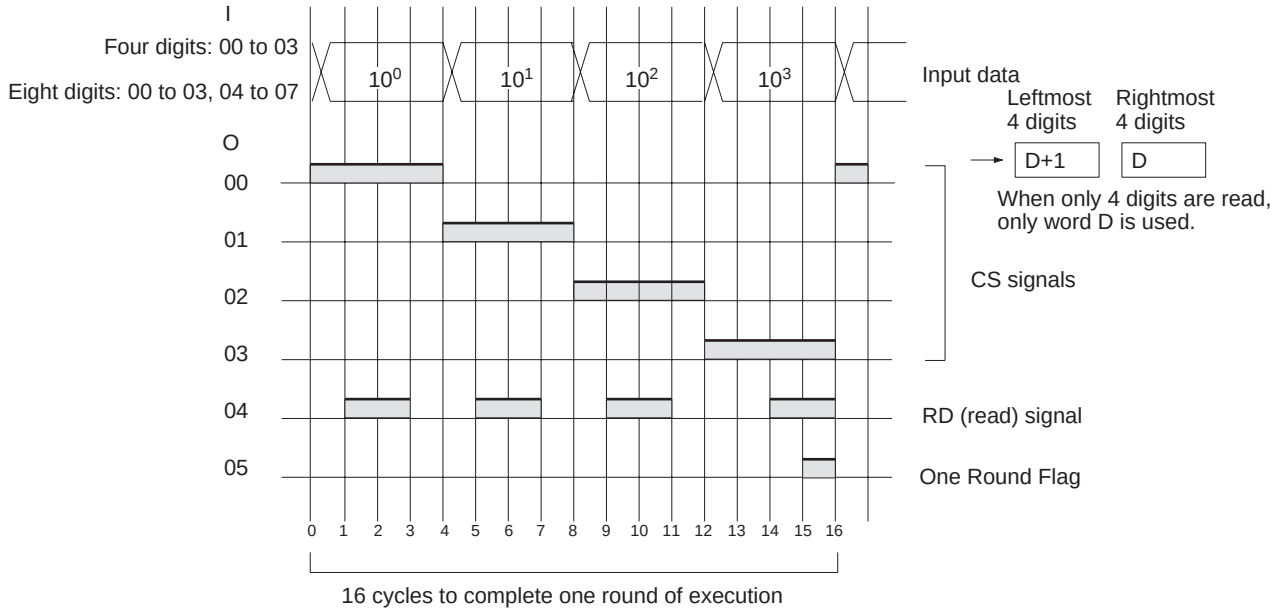
There is no restriction on the number of times that DSW(210) can appear in the program.

External Connections

Connect the digital switch or thumbwheel switch to Input Unit contacts 0 to 7 and Output Unit contacts 0 to 4, as shown in the following diagram. The following example illustrates connections for an A7B Thumbwheel Switch.



Timing Chart



Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

Do not read or write the system word (C2) from any other instruction. DSW(210) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by DSW(210) in the first cycle when program execution starts. If DSW(210) is being used from the first cycle, clear the system word from the program.

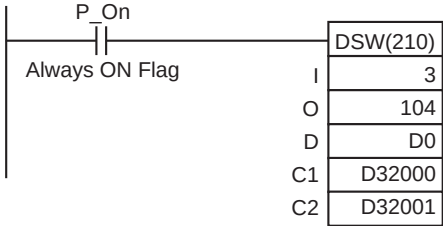
DSW(210) will not operate correctly if I/O refreshing is not performed with the Input Unit and Output Unit connected to the digital switch or thumbwheel switch after DSW(210) is executed. Consequently, do not connect the digital switch or thumbwheel switch to the following Units.

- Communications Slaves (DeviceNet or CompoBus/S Slaves)

Example

In this example, DSW(210) is used to read an 8-digit number from a digital switch and outputs the resulting value constantly to D0 to D3. The digital switch is connected through CIO 3 and CIO 104.

Since 8 digits of data are being read, C1 (D32000 in this case) is set to 0001 hex. D32001 is used as the system word.



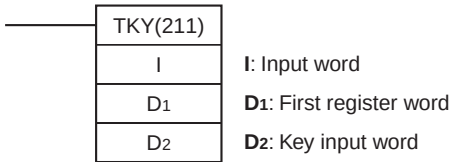
3-22-4 TEN KEY INPUT – TKY(211)

Purpose

Reads numeric data from a ten-key keypad and stores up to 8 digits of BCD data in the specified words.

A Unit with 10 inputs or more is required for this instruction.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	TKY(211)
	Executed Once for Upward Differentiation	@TKY(211)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

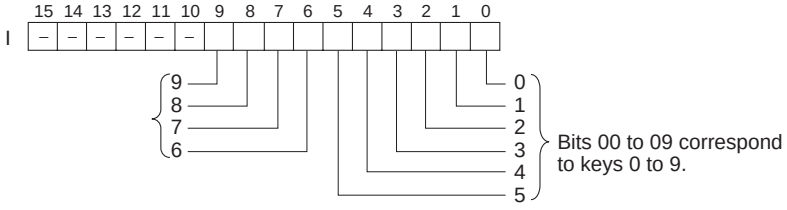
Applicable Program Areas

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

Operands

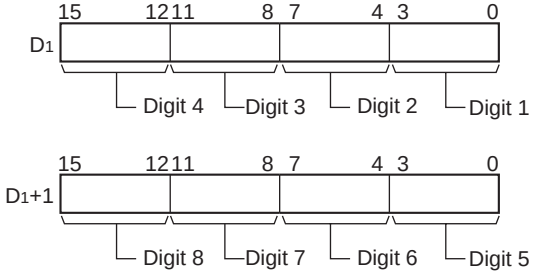
I: Input Word (Data Line Inputs)

Specify the input word allocated to the Input Unit and connect the ten-key keypad's 0 to 9 data lines to the Input Unit as shown in the following diagram.



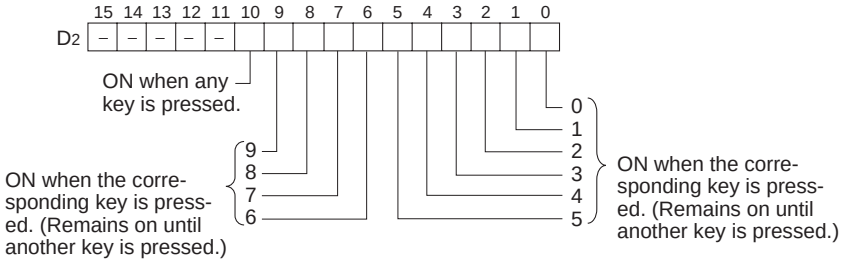
D_1: First Register Word

Specifies the leading word address where the ten-key keypad's numeric input (up to 8 digits) will be stored.



D_2: Key Input Word

Bits 00 to 10 of D_2 indicate key inputs. When one of the keys on the keypad (0 to 9) has been pressed, the corresponding bit of D_2 (0 to 9) is turned ON. Bit 10 of D_2 will be ON while any key is being pressed.



Operand Specifications

Area	I	D_1	D_2
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A959	A448 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

TKY(211) reads numeric data from input word I, which is allocated to a ten-key keypad connected to an Input Unit, and stores up to 8 digits of BCD data in register words D_1 and D_1+1. In addition, each time that a key is pressed, the corresponding bit in D_2 (0 to 9) will be turned ON and remains ON until another key is pressed. Bit 10 of D_2 will be ON while any key is being pressed and OFF when no key is being pressed.

The two-word register in D_1 and D_1+1 operates as an 8-digit shift register. When a key is pressed on the ten-key keypad, the corresponding BCD digit is shifted into the least significant digit of D_1. The other digits of D_1, D_1+1 are shifted left and the most significant digit of D_1+1 is lost.

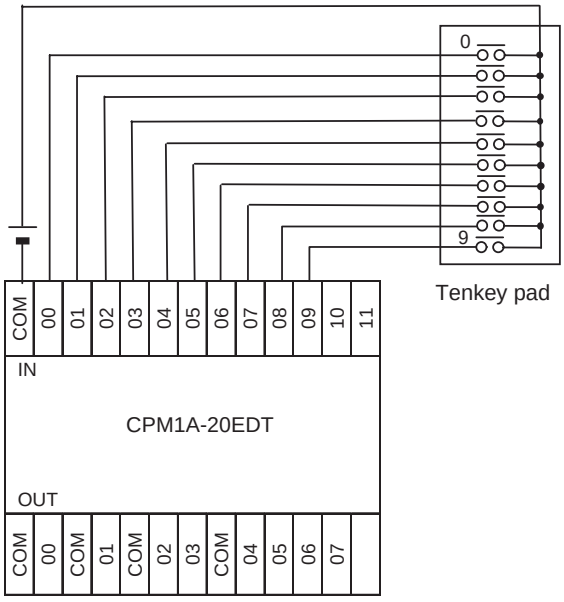
When executed, TKY(211) begins reading the key input data from the first cycle, regardless of the point at which the last instruction was stopped.

When one of the keypad keys is being pressed, input from the other keys is disabled.

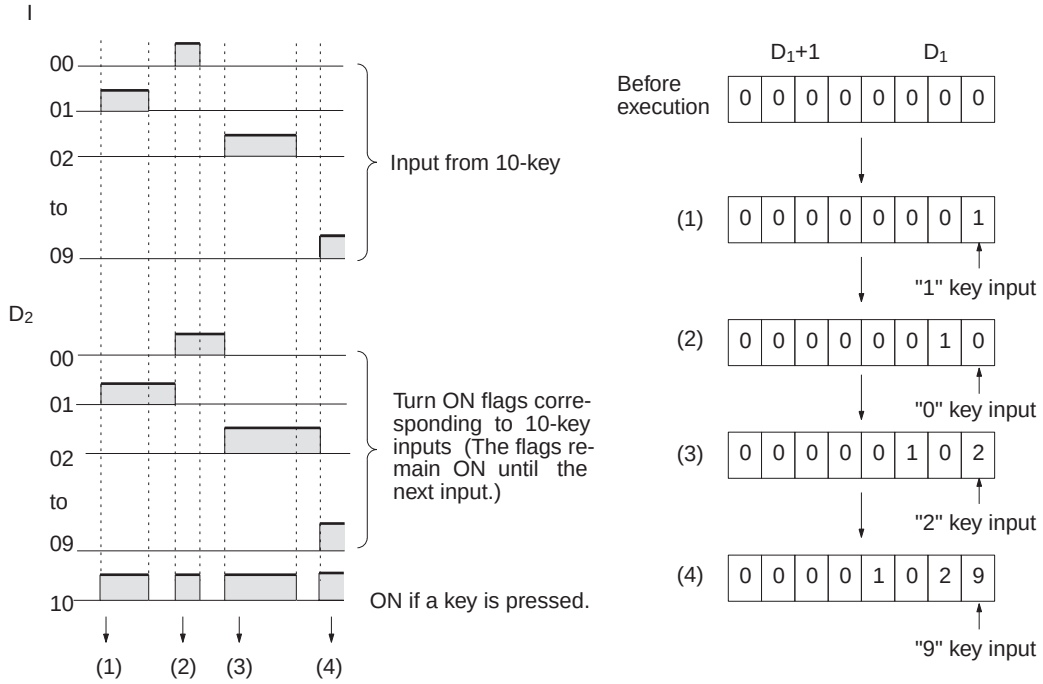
There is no restriction on the number of times that TKY(211) can appear in the program.

External Connections

Connect the ten-key keypad so that the switches for keys 0 through 9 are input to contacts 0 through 9 of the Input Unit, as shown in the following diagram.



Timing Chart



Flags

Name	Label	Operation
Error Flag	ER	OFF

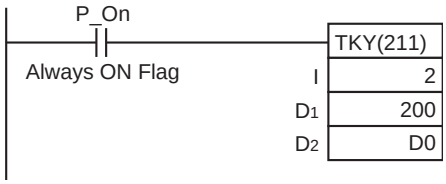
Precautions

TKY(211) will not operate correctly if I/O refreshing is not performed with the Input Unit connected to the ten-key keypad after TKY(211) is executed. Consequently, do not connect the ten-key keypad to the following Units.

- Communications Slaves (DeviceNet or CompoBus/S Slaves)

Example

In this example, TKY(211) reads key inputs from a ten-key keypad and stores the inputs in CIO 200 and CIO 201. The ten-key keypad is connected to CIO 2.

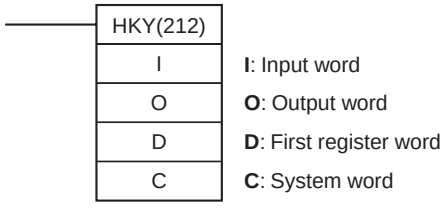


3-22-5 HEXADECIMAL KEY INPUT – HKY(212)

Purpose

Reads numeric data from a hexadecimal keypad connected to an Input Unit and Output Unit and stores up to 8 digits of hexadecimal data in the specified words.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	HKY(212)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

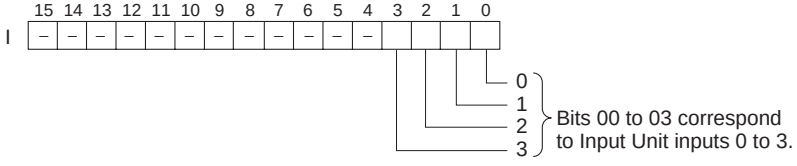
Applicable Program Areas

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

Operands

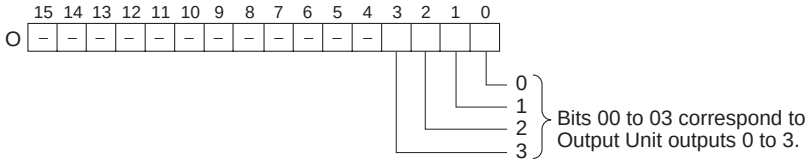
I: Input Word (Data Line D0 to D3 Inputs)

Specify the input word allocated to the Input Unit and connect the hexadecimal keypad's D0 to D3 data lines to the Input Unit as shown in the following diagram.



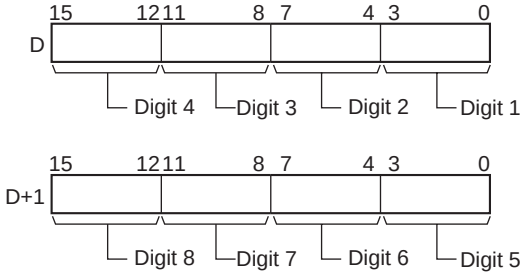
O: Output Word (Selection Signal Outputs)

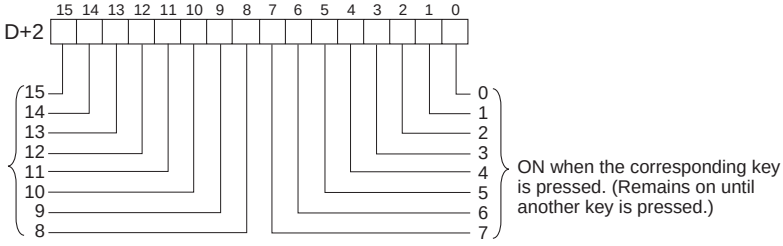
Specify the output word allocated to the Output Unit and connect the hexadecimal keypad's selection signals to the Output Unit as shown in the following diagram.



D: First Register Word

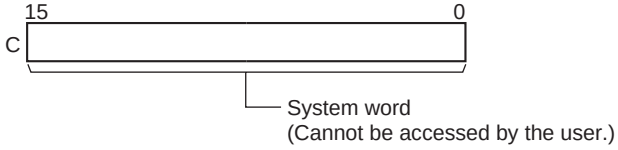
Specifies the leading word address where the hexadecimal keypad's numeric input (up to 8 digits) will be stored. (In addition, each time that a key is pressed, the corresponding bit in D+2 (0 to F) will be turned ON and remains ON until another key is pressed.)





C: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	I	O	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6141	CIO 0 to CIO 6143
Work Area	W0 to W511		W0 to W509	W0 to W511
Holding Bit Area	H0 to H511		H0 to H509	H0 to H511
Auxiliary Bit Area	A0 to A957	A448 to A959	A448 to A957	A448 to A959
Timer Area	T0000 to T4095		T0000 to T4093	T0000 to T4095
Counter Area	C0000 to C4095		C0000 to C4093	C0000 to C4095

Area	I	O	D	C
DM Area	D0 to D32767		D0 to D32765	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767			
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	---			
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

HKY(212) outputs the selection signals to bits 00 to 03 of O, reads the data in order from bits 00 to 03 of I, and stores up to 8 digits of hexadecimal data in register words D and D+1.

HKY(212) inputs each digit in 3 to 12 cycles, and then starts over and continues inputting. In addition, each time that a key is pressed, the corresponding bit in D+2 (0 to F) will be turned ON and remains ON until another key is pressed.

HKY(212) determines which key is pressed by identifying which input is ON when a given selection signal is ON, so it can take anywhere from 3 to 12 cycles for one hexadecimal digit to be read. After the key input is read, HKY(212) starts over and reads another digit in the next 3 to 12 cycles.

When executed, HKY(212) begins reading the key input data from the first selection signal, regardless of the point at which the last instruction was stopped.

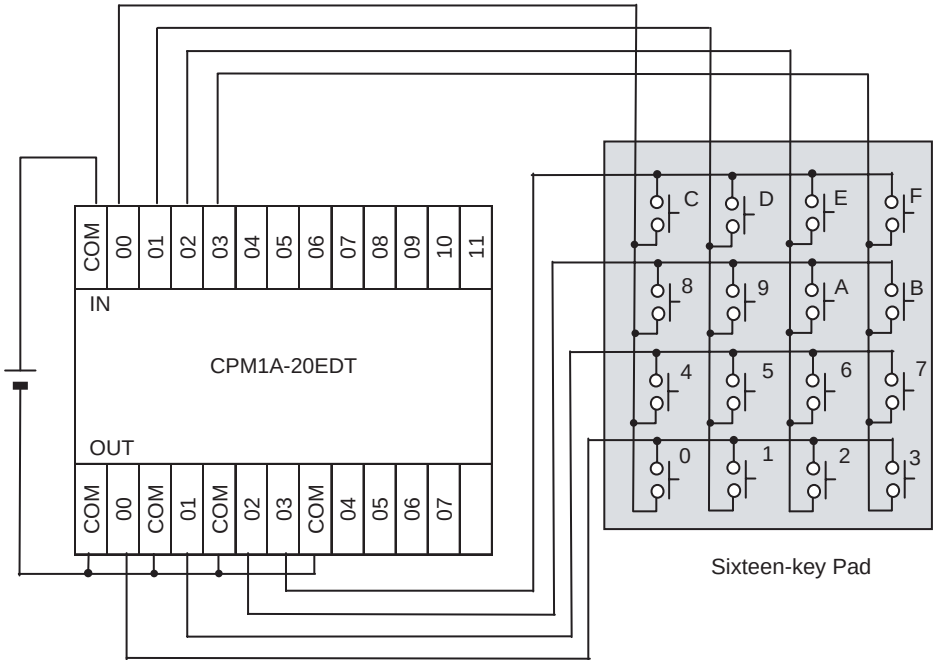
The two-word register in D_1 and D_1+1 operates as an 8-digit shift register. When a key is pressed on the ten-key keypad, the corresponding hexadecimal digit is shifted into the least significant digit of D_1. The other digits of D_1, D_1+1 are shifted left and the most significant digit of D_1+1 is lost.

When one of the keypad keys is being pressed, input from the other keys is disabled.

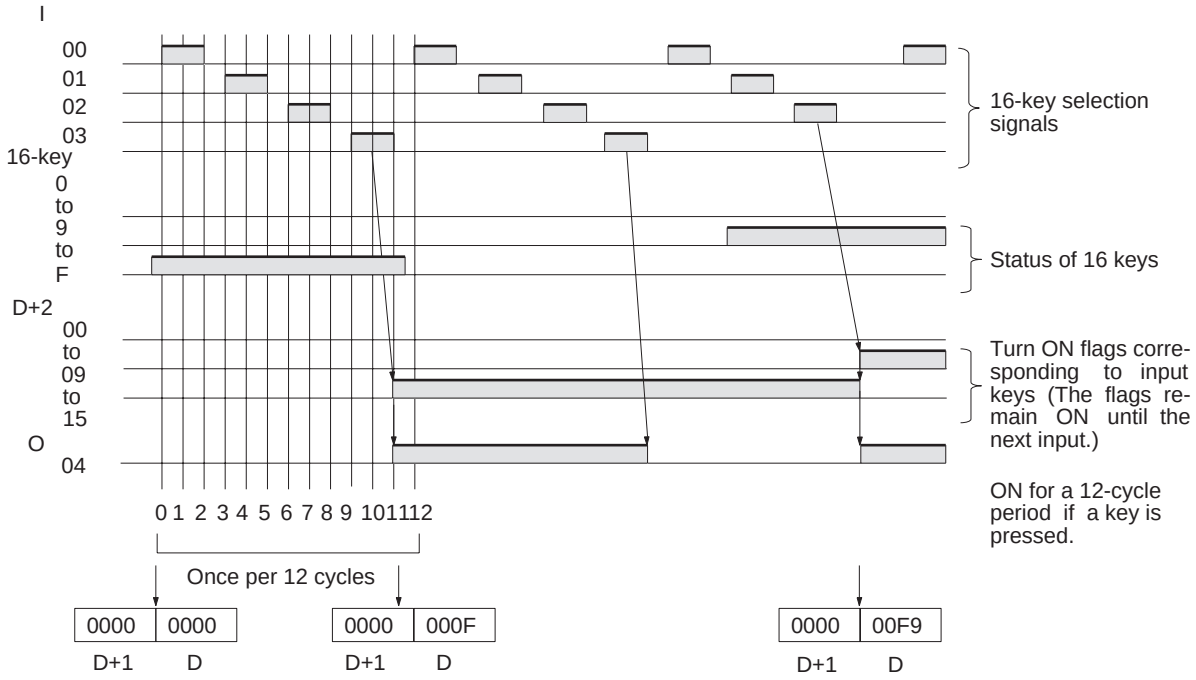
There is no restriction on the number of times that HKY(212) can appear in the program.

External Connections

Connect the hexadecimal keypad to Input Unit contacts 0 to 3 and Output Unit contacts 0 to 3, as shown in the following diagram.



Timing Chart



Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

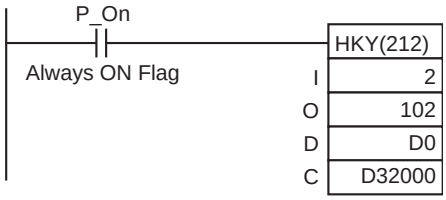
Do not read or write the system word (C) from any other instruction. HKY(212) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by HKY(212) in the first cycle when program execution starts. If HKY(212) is being used from the first cycle, clear the system word from the program.

HKY(212) will not operate correctly if I/O refreshing is not performed with the Input Unit and Output Unit connected to the hexadecimal keypad after HKY(212) is executed. Consequently, do not connect the hexadecimal keypad to the following Units.

- Communications Slaves (DeviceNet or CompoBus/S Slaves)

Example

In this example, HKY(212) reads up to 8 digits of hexadecimal data from a hexadecimal keypad and stores the data in D0 and D1. The hexadecimal keypad is connected through CIO 2 and CIO 102. D32000 is used as the system word.

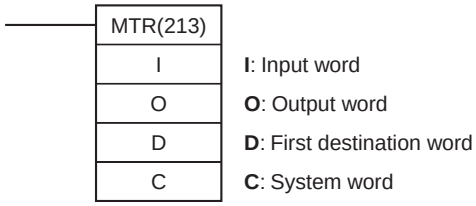


3-22-6 MATRIX INPUT: MTR(213)

Purpose

Inputs up to 64 signals from an 8 × 8 matrix connected to an Input Unit and an Output Unit (using 8 input points and 8 output points) and stores that 64-bit data in the 4 destination words.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MTR(213)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

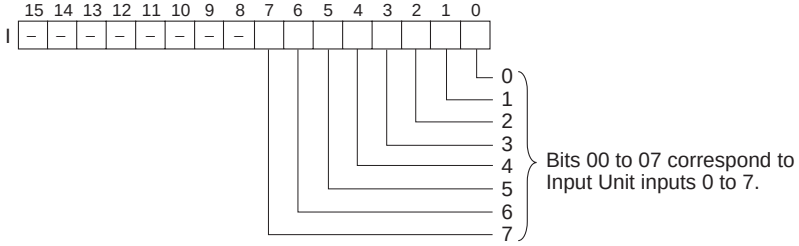
Applicable Program Areas

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

Operands

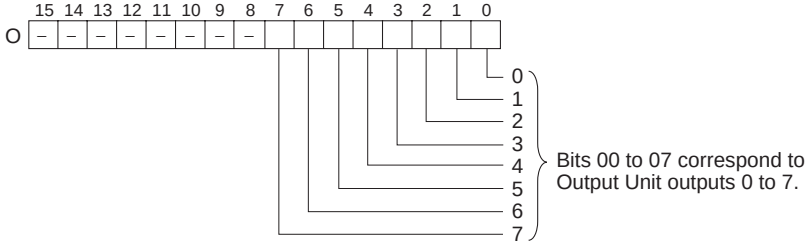
I: Input Word

Specify the input word allocated to the Input Unit and connect the 8 input signal lines to the Input Unit as shown in the following diagram.



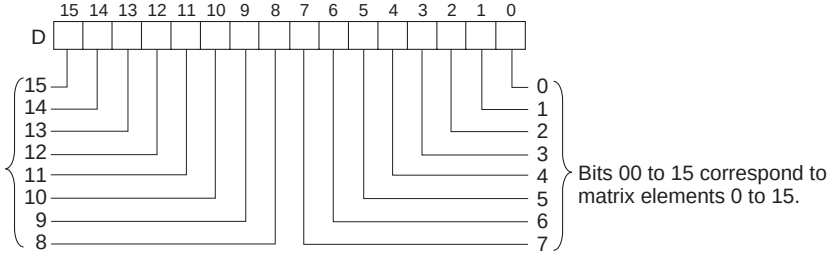
O: Output Word (Selection Signal Outputs)

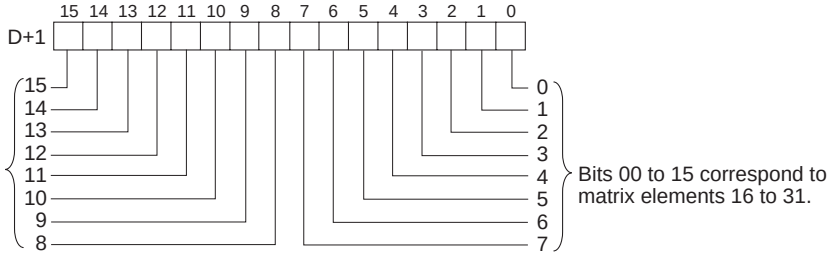
Specify the output word allocated to the Output Unit and connect the 8 selection signals to the Output Unit as shown in the following diagram.

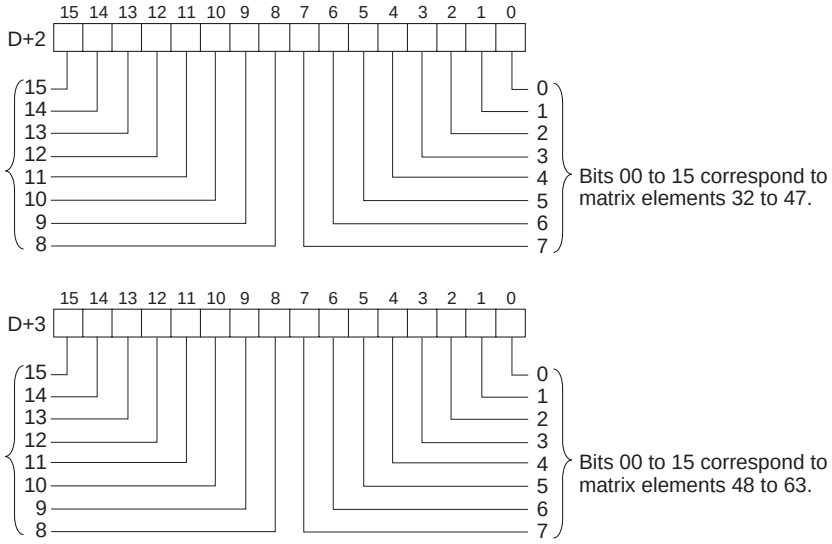


D: First Register Word

Specifies the leading word address of the 4 words that contain the data from the 8 × 8 matrix.

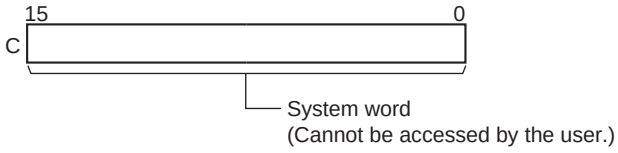






C: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	I	O	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6140	CIO 0 to CIO 6143
Work Area	W0 to W511		W0 to W508	W0 to W511
Holding Bit Area	H0 to H511		H0 to H508	H0 to H511
Auxiliary Bit Area	A0 to A959	A448 to A959	A448 to A956	A448 to A959
Timer Area	T0000 to T4095		T0000 to T4092	T0000 to T4095
Counter Area	C0000 to C4095		C0000 to C4092	C0000 to C4095
DM Area	D0 to D32767		D0 to D32764	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767			
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	---			
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

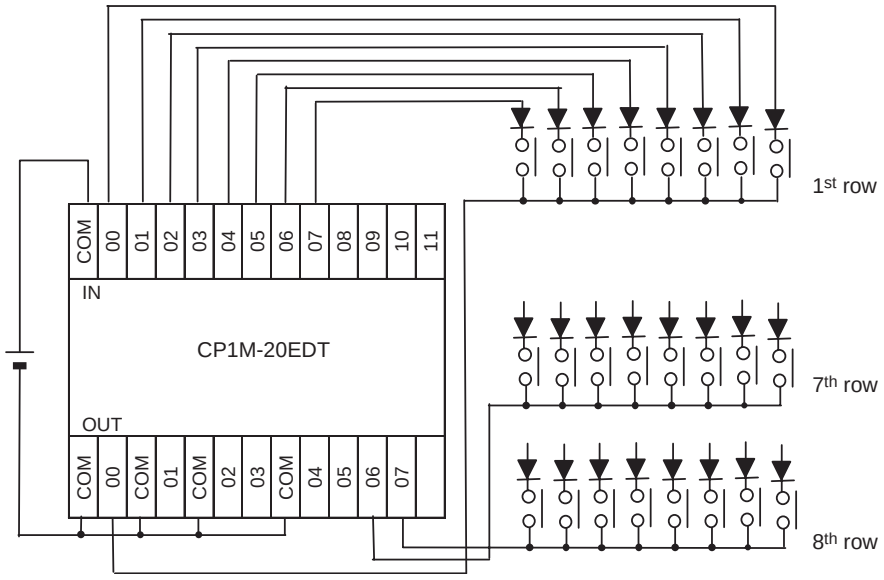
MTR(213) outputs the selection signals to bits 00 to 07 of O, reads the data in order from bits 00 to 07 of I, and stores the 64 bits of data in the 4 words D through D+3. MTR(213) reads the status of the 64-bit matrix every 24 CPU Unit cycles. The One Round Flag (bit 08 of O) is turned ON for one cycle in every 24 cycles after each of the selection signals has been turned ON.

When executed, MTR(213) begins reading the matrix status from the beginning of the matrix, regardless of the point at which the last instruction was stopped.

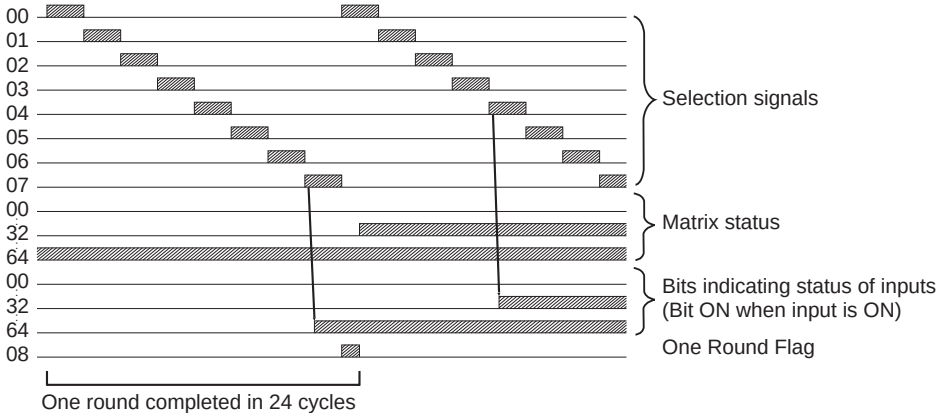
There is no restriction on the number of times that MTR(213) can appear in the program.

External Connections

Connect the hexadecimal keypad to Input Unit contacts 0 to 3 and Output Unit contacts 0 to 3, as shown in the following diagram.



Timing Chart



Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

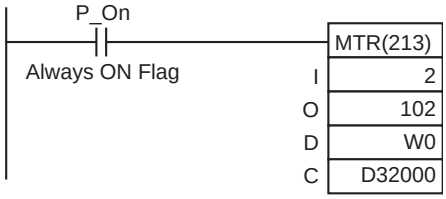
Do not read or write the system word (C) from any other instruction. MTR(213) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by MTR(213) in the first cycle when program execution starts. If MTR(213) is being used from the first cycle, clear the system word from the program.

MTR(213) will not operate correctly if I/O refreshing is not performed with the Input Unit and Output Unit connected to the external matrix after MTR(213) is executed. Consequently, do not connect the external matrix to the following Units.

- Communications Slaves (DeviceNet or CompoBus/S Slaves)

Example

In this example, MTR(213) reads the 64 bits of data from the 8 × 8 matrix and stores the data in W0 to W3. The 8 × 8 matrix is connected through CIO 2 and CIO 102. D32000 is used as the system word.

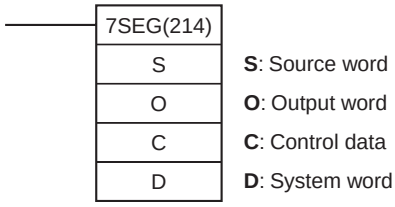


3-22-7 7-SEGMENT DISPLAY OUTPUT – 7SEG(214)

Purpose

Converts the source data (either 4-digit or 8-digit BCD) to 7-segment display data, and outputs that data to the specified output word.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	7SEG(214)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

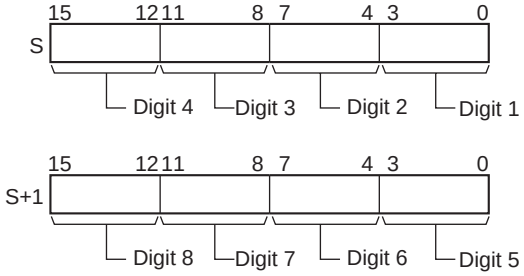
Applicable Program Areas

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

Operands

S: Source Word

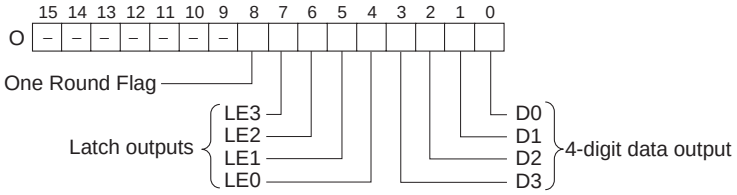
Specify the first source word containing the data that will be converted to 7-segment display data.



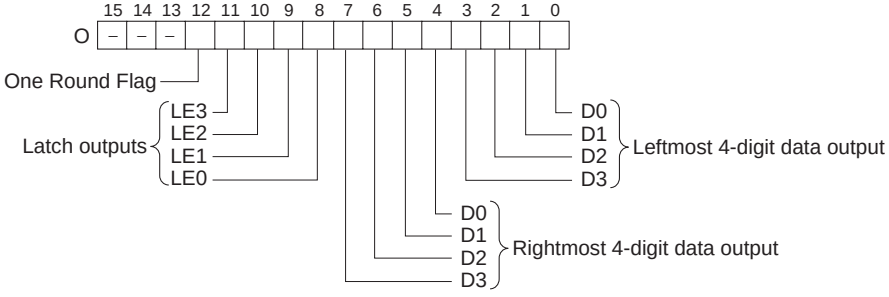
O: Output Word (Data and Latch Outputs)

Specify the output word allocated to the Output Unit and connect the 7-segment display to the Output Unit as shown in the following diagram.

- Converting 4 digits



- Converting 8 digits



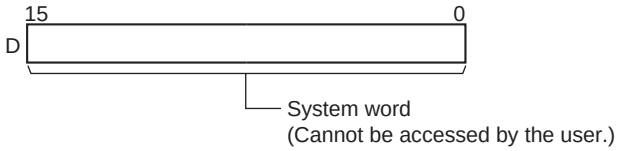
C: Control Data

The value of C indicates the number of digits of source data and the logic for the Input and Output Units, as shown in the following table. (The logic refers to the transistor output's NPN or PNP logic.)

Source data	Display's data input logic	Display's latch input logic	C
4 digits (S)	Same as Output Unit	Same as Output Unit	0000
		Different from Output Unit	0001
	Different from Output Unit	Same as Output Unit	0002
		Different from Output Unit	0003
8 digits (S, S+1)	Same as Output Unit	Same as Output Unit	0004
		Different from Output Unit	0005
	Different from Output Unit	Same as Output Unit	0006
		Different from Output Unit	0007

D: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	S	O	C	D
CIO Area	CIO 0 to CIO 6143		---	CIO 0 to CIO 6143
Work Area	W0 to W511		---	W0 to W511
Holding Bit Area	H0 to H511		---	H0 to H511
Auxiliary Bit Area	A0 to A959	A448 to A959	---	A448 to A959
Timer Area	T0000 to T4095		---	T0000 to T4095
Counter Area	C0000 to C4095		---	C0000 to C4095
DM Area	D0 to D32767		---	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		---	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	---	---	0000 to 0007	---
Data Registers	---	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to ,–(– –)IR15		---	,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

7SEG(214) reads the source data, converts it to 7-segment display data, and outputs that data (as leftmost 4 digits D0 to D3, rightmost 4 digits D0 to D3, latch output signals LE0 to LE3) to the 7-segment display connected to the output indicated by O. The value of C indicates the number of digits of source data (either 4-digit or 8-digit) and the logic for the Input and Output Units.

7SEG(214) displays the 4-digit or 8-digit data in 12 cycles, and then starts over and continues displaying the data.

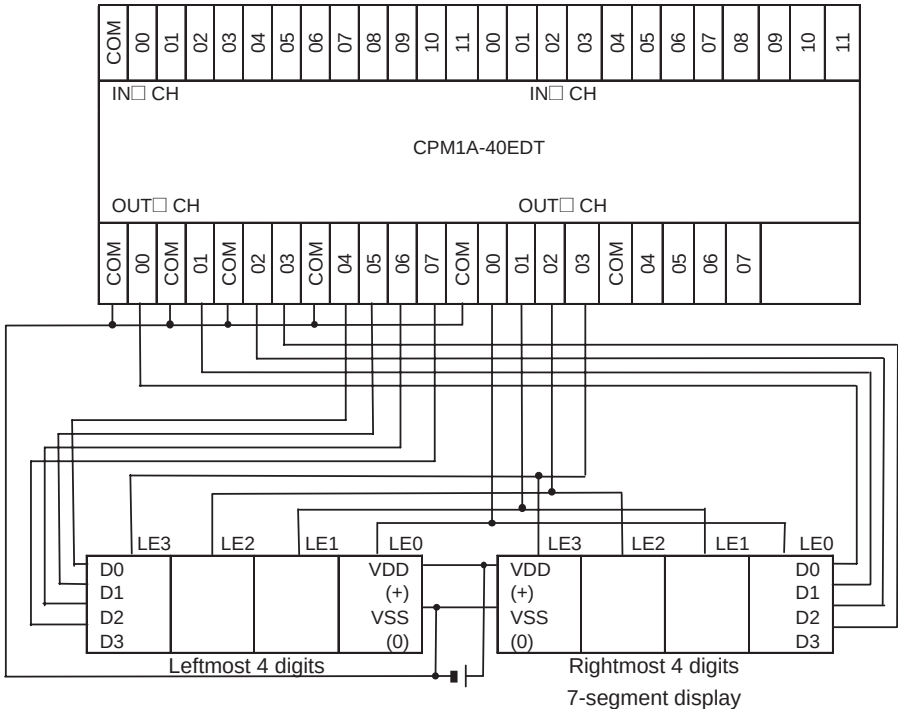
The One Round Flag (bit 08 of O when converting 4 digits, bit 12 of O when converting 8 digits) is turned ON for one cycle in every 12 cycles after 7SEG(214) has turned ON each of the latch output signals. After the 7-segment data is output in 12 cycles, 7SEG(214) starts over and converts the present contents of the source word(s) in the next 12 cycles.

When executed, 7SEG(214) begins on latch output 0 at the beginning of the round, regardless of the point at which the last instruction was stopped.

Even if the connected 7-segment display has fewer than 4 digits or 8 digits in its display, 7SEG(214) will still output 4 digits or 8 digits of data.

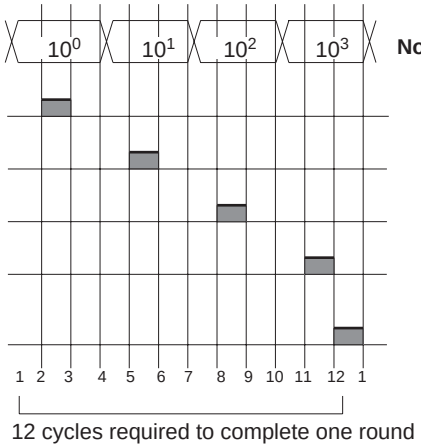
External Connections

Connect the 7-segment display to the Output Unit as shown in the following diagram. This example shows an 8-digit display. With a 4-digit display, the data outputs (D0 to D3) would be connected to outputs 0 to 3 and the latch outputs (LE0 to LE3) would be connected to outputs 4 to 7. Output point 12 (for 8-digit display) or output point 8 (for 4-digit display) will be turned ON when one round of data has been output, but it is not necessary to connect them unless required by the application.



Timing Chart

Function	Bit(s) in O		Output status (Data and latch logic depends on C)
	(4 digits, 1 block)	(4 digits, 2 blocks)	
Data output	00 to 03	00 to 03 04 to 07	
Latch output 0	04	08	
Latch output 1	05	09	
Latch output 2	06	10	
Latch output 3	07	11	
One Round Flag	08	12	



Note 0 to 3: Data output for word S
4 to 7: Data output for word S+1

Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

Do not read or write the system word (D) from any other instruction. 7SEG(214) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by 7SEG(214) in the first cycle when program execution starts. If 7SEG(214) is being used from the first cycle, clear the system word from the program.

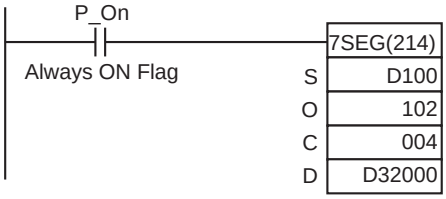
7SEG(214) will not operate correctly if I/O refreshing is not performed with the Output Unit connected to the 7-segment display after 7SEG(214) is executed. Consequently, do not connect the external matrix to the following Units.

- Communications Slaves (DeviceNet or CompoBus/S Slaves)

Example

In this example, 7SEG(214) converts the 8 digits of BCD data in D100 and D101 and outputs the data through CIO 102.

There are 8 digits of data being output and the 7-segment display's logic is the same as the Output Unit's logic, so the control data (C) is set to 0004. D32000 is used as the system word, D.

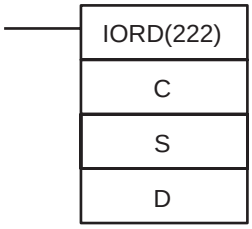


3-22-8 INTELLIGENT I/O READ: IORD(222)

Purpose

Reads the contents of memory area of a Special I/O Unit or CPU Bus Unit.

Ladder Symbol



C: Control data

S: Transfer source and number of words

D: Transfer destination

Variations

Variations		
	Executed Each Cycle for ON Condition	IORD(222)
	Executed Once for Upward Differentiation	@IORD(222)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

C: Depends on Special I/O Unit or CPU Bus Unit.
S: Special I/O Unit: 0000 to 005F hex
(to specify unit numbers 0 to 95)
CPU Bus Unit: 8000 to 800F hex
(to specify unit numbers 0 to F hex)
S+1: Number of words to transfer
(0001 to 0080 hex, depends on Special I/O Unit or CPU Bus Unit)

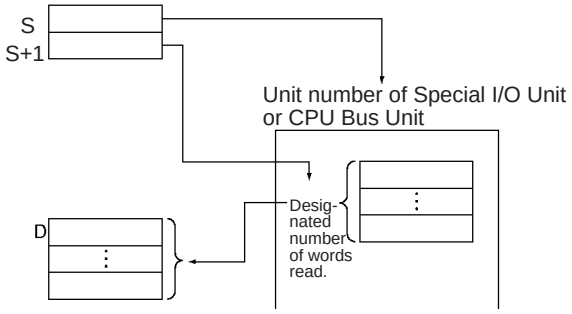
S+1: Leftmost 4 digits
S: Rightmost 4 digits

Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	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

IORD(222) reads the number of words designated in S+1 from the memory area of the Special I/O Unit or CPU Bus Unit whose unit number is designated by S and outputs the data to D. Refer to the operation manual of the Special I/O Unit or CPU Bus Unit from which data is being read for specific details for each Unit.



Flags

Name	Label	Operation
Error Flag	ER	ON if the number of words to transfer (S) is outside the range of 0001 to 0080 hex. ON if the unit number (S) is outside the range of 0000 to 005F hex or 8000 to 800F hex. ON if a Special I/O Unit or CPU Bus Unit not affected by IORD(222) is designated. ON if a Special I/O Unit with a setting error or an error is designated. ON if a CPU Bus Unit with a setting error or an error is designated. OFF in all other cases.
Equals Flag	=	ON if reading operation is completed normally. OFF if reading operation is not completed normally.

Precautions

The Equals Flag will turn ON if the reading operation is completed normally.

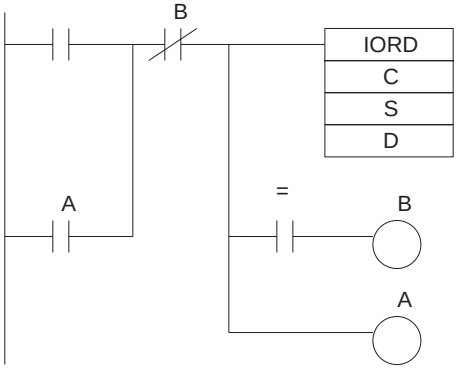
The Equals Flag will turn OFF if the reading operation cannot be completed normally due to the Special I/O Unit or CPU Bus Unit being busy.

Whenever any of the following occur, an error will occur and the Error Flag will turn ON.

- The number of words to transfer (S) is outside the range of 0001 to 0080 (hex).
- The unit number (S) is outside the range of 0000 to 005F hex or 8000 to 800F hex.
- A Special I/O Unit or CPU Bus Unit not affected by IORD(222) is designated.
- A Special I/O Unit with a setting error or an error is designated.
- A CPU Bus Unit with a setting error or error is designated.

When IORD(222) is executed, the execution results are reflected in the condition flags. In particular, the Equals Flag turns ON when reading is completed. Input the condition flags such as the Equals Flag with output branching from the same input conditions as the IORD(222) instruction.

If the Special I/O Unit or CPU Bus Unit is busy, the reading operation will not be executed. Use the Equals Flag to create a self-maintaining program, as shown below, so that IORD(222) will be executed with each cycle until the reading operation is executed.

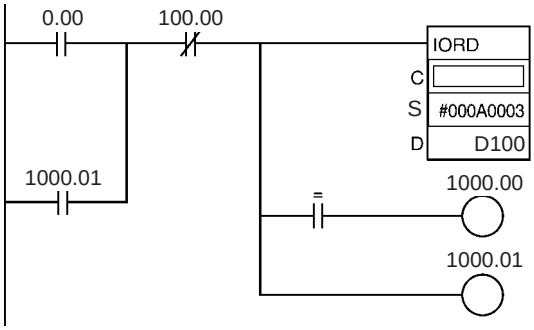


When the input condition is met, self maintenance is performed by output A and IORD(222) is executed with each cycle until the Equals Flag turns ON. When the reading is completed and the Equals Flag turns ON, output B turns ON and the self maintenance is cleared.

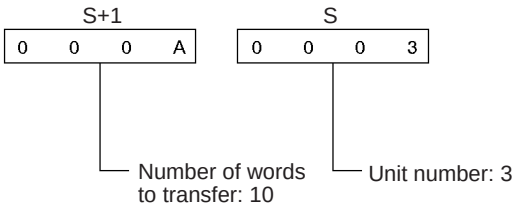
Be sure to place condition flags directly after IORD(222) instructions, and not after any other instructions. If a condition flag is placed after another instruction, it will be affected by the execution results of that instruction.

Example

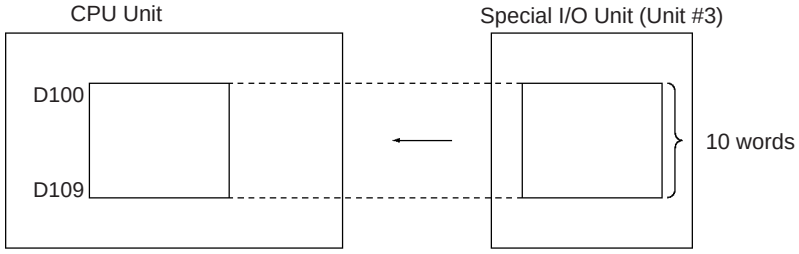
In this example, IORD(222) is used to read data.



When CIO 0.00 is turned ON, 10 words are read from the Special I/O Unit with unit number 3, and are stored in D100 to D109.



The control code (C) varies depending on the Special I/O Unit.

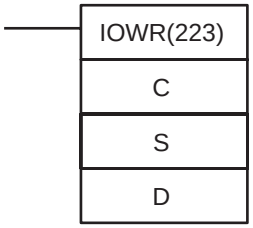


3-22-9 INTELLIGENT I/O WRITE: IOWR(223)

Purpose

Outputs the contents of the CPU Unit's I/O memory area to a Special I/O Unit or CPU Bus Unit.

Ladder Symbol



C: Control data

S: Transfer source and number of words

D: Transfer destination and number of words

Variations

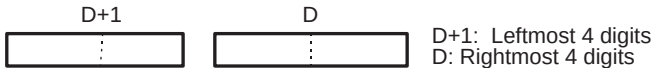
Variations	**Executed Each Cycle for ON Condition**	IOWR(223)
	Executed Once for Upward Differentiation	@IOWR(223)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

C: Depends on Special I/O Unit or CPU Bus Unit.
D: Special I/O Unit: 0000 to 005F hex
(to specify unit numbers 0 to 95)
CPU Bus Unit: 8000 to 800F hex
(to specify unit numbers 0 to F hex)
D+1: Number of words to transfer
(0000 to 0080 hex, depends on Special I/O Unit or CPU Bus Unit)

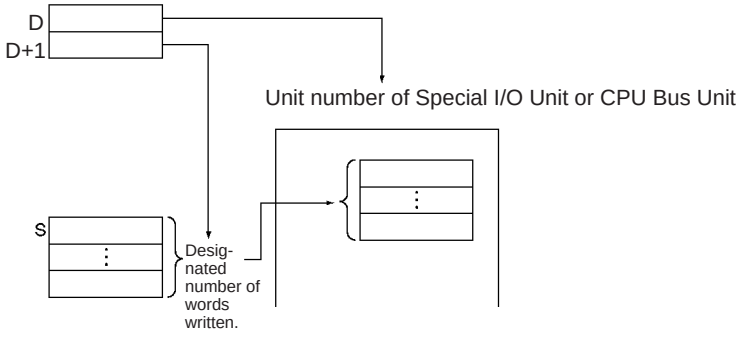


Operand Specifications

Area	C	S	D
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6142
Work Area	W0 to W511		W0 to W510
Holding Bit Area	H0 to H511		H0 to H510
Auxiliary Bit Area	A0 to A959		A0 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D0 to D32767		D0 to D32766
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)		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

IOWR(223) writes the designated number of words (D) from the first source word (designated by S) onwards and outputs them to the Special I/O Unit or CPU Bus Unit that has the unit number designated by D.



Flags

Name	Label	Operation
Error Flag	ER	ON if the number of words to transfer (D) is outside the range of 0001 to 0080 hex. ON if the unit number (D) is outside the range of 0000 to 005F hex or 8000 to 800F hex. ON if S is designated by a constant when the number of words to be transferred (D+1) is not 0001 hex. ON if a Special I/O Unit or CPU Bus Unit not affected by IOWR(223) is designated. ON if a Special I/O Unit with a setting error or an error is designated. ON if a CPU Bus Unit with a setting error or an error is designated. OFF in all other cases.
Equals Flag	=	ON if writing operation is completed normally. OFF if writing operation is not completed normally.

Precautions

When “0001” is designated for the number of words to be transferred (D+1), the data for S can be designated by a constant. If a constant is designated for S when the number of words to be transferred is not “0001,” an error will occur and the Error Flag will turn ON.

The Equals Flag will turn ON if the writing operation is completed normally.

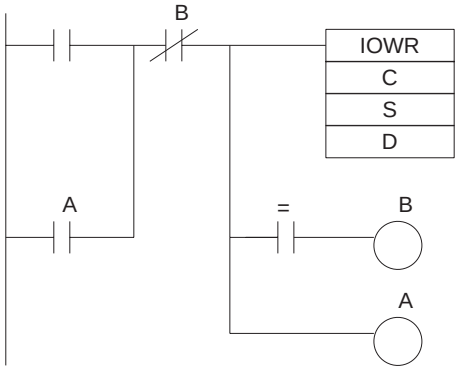
The Equals Flag will turn OFF if the writing operation cannot be completed normally due to the Special I/O Unit or CPU Bus Unit being busy.

Whenever any of the following occur, an error will occur and the Error Flag will turn ON.

- There is an I/O Unit verification error, a Special I/O Unit setting error, a Special I/O Unit setting error, or a Special I/O Unit error at the Special I/O Unit.
- There is an I/O Unit verification error, a CPU Bus Unit setting error, a CPU Bus Unit setting error, or a CPU Bus Unit error at the CPU Bus Unit.
- The number of words to transfer (D) is outside the range of 0001 to 0080 (hex).
- The unit number (D) is outside the range of 0000 to 005F hex or 8000 to 800F hex.
- A Special I/O Unit or CPU Bus Unit not affected by IOWR(223) is designated.
- A Special I/O Unit with a setting error or an error is designated.
- A CPU Bus Unit with a setting error or an error is designated.

When IOWR(223) is executed, the execution results are reflected in the condition flags. In particular, the Equals Flag turns ON when reading is completed. Input the condition flags such as the Equals Flag with output branching from the same input conditions as the IOWR(223) instruction.

If the Special I/O Unit or CPU Bus Unit is busy, the writing operation will not be executed. Use the Equals Flag to create a self-maintaining program, as shown below, so that IOWR(223) will be executed with each cycle until the writing operation is executed.

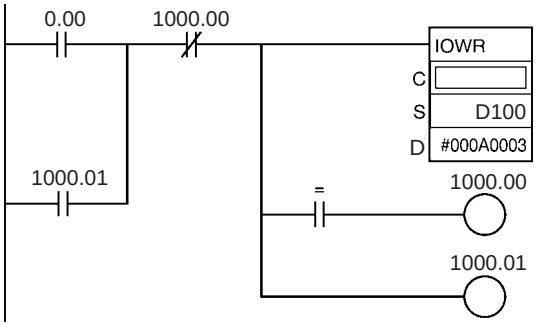


When the input condition is met, self maintenance is performed by output A and IOWR(223) is executed with each cycle until the Equals Flag turns ON. When the writing is completed and the Equals Flag turns ON, output B turns ON and the self maintenance is cleared.

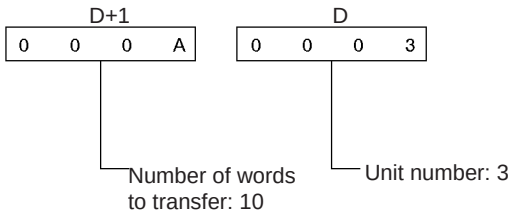
Be sure to place condition flags directly after IOWR(223) instructions, and not after any other instructions. If a condition flag is placed after another instruction, it will be affected by the execution results of that instruction.

Example

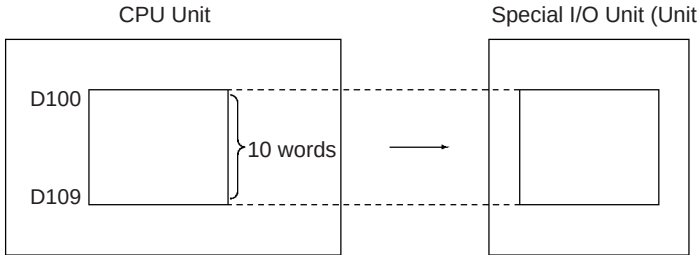
In this example, IOWR(223) is used to write data.



When CIO 0.00 is turned ON, the 10 words in D100 to D109 are written to the Special I/O Unit.



The control code (C) varies depending on the Special I/O Unit.



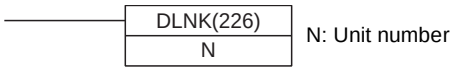
3-22-10 CPU BUS UNIT I/O REFRESH: DLNK(226)

Purpose

Performs I/O refreshing immediately for the CPU Bus Unit with the specified unit number. The following data is refreshed:

- The words allocated to the CPU Bus Unit in the PLC's CPU Bus Unit Areas (25 words in the CIO Area and 100 words in the DM Area)
- Specific data refreshing for Units such as Units that support data links

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	DLNK(226)
	Executed Once for Upward Differentiation	@DLNK(226)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Unit number

Specifies the CPU Bus Unit's unit number (0000 to 000F hex or 0 to 15 decimal).

Operand Specifications

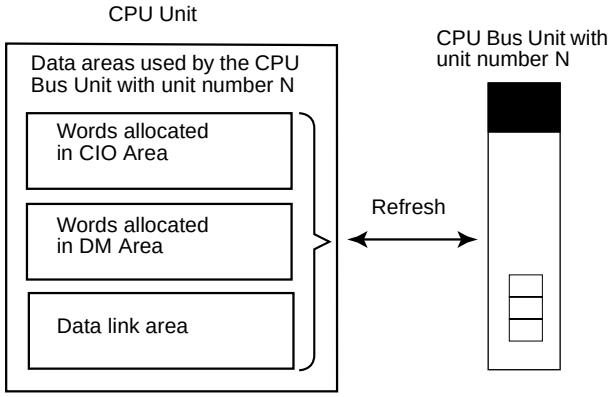
Area	N
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	#0000 to #000F (binary) or 0 to 15 (decimal)
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

DLNK(226) performs immediate I/O refreshing for the CPU Bus Unit with the specified unit number. The data listed below is refreshed. Refer to the *Precautions* below for details on the execution conditions to use for immediate refreshing.

1. The words allocated to the CPU Bus Unit in the PLC's CPU Bus Unit Areas (25 words in the CIO Area and 100 words in the DM Area)
2. Data specific the CPU Bus Unit such as data link data or DeviceNet Remote I/O Communications data (refreshed together with the data in the CPU Bush Unit Areas)

CPU Bus Unit	Data refreshing specific to the Unit
Controller Link Unit	Data link refreshing
DeviceNet Unit	Remote I/O communications refreshing



The following table shows how DLNK(226) differs from IORF(097).

Instruction	Operation
DLNK(226)	• I/O refreshing of the CPU Bus Unit Area in the CIO Area (25 words) • I/O refreshing of the CPU Bus Unit Area in the DM Area (100 words) • Refreshing of data specific to the CPU Bus Unit, such as data link data or DeviceNet Remote I/O Communications data
IORF(097)	• I/O refreshing of words used by CPM1A Expansion I/O Units or CPM1A Expansion Units • I/O refreshing of the 10 CIO words allocated to a Special I/O Unit

DLNK(226) refreshes data between the CPU Unit and specified CPU Bus Unit. There are two special factors to consider when using DLNK(226):

1,2,3...

1. When exchanging data through a data link or DeviceNet remote I/O communications, the data exchange is not performed with the other Units at the same time that DLNK(226) is executed. The data exchange will be performed when the network communications cycle reaches the Unit in question and data is exchanged with that Unit. Consequently, the actual data exchange may be delayed by as much as the communications cycle time of the network.
2. DLNK(226) cannot perform I/O refreshing with a CPU Bus Unit if that Unit is currently exchanging data. If DLNK(226) is executed too frequently, I/O refreshing will not be performed. We recommend allowing a delay between executions of DLNK(226) that is longer than the communications cycle time.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified unit number is not between 0000 and 000F hex (between 0 and 15 decimal). ON if the PLC does not have a CPU Bus Unit with the specified unit number. OFF in all other cases.
Equals Flag	=	OFF if the I/O refreshing could not be performed because the CPU Bus Unit was refreshing data. OFF if there was an error or setting error in the specified CPU Bus Unit. OFF if DLNK(226) was executed in an interrupt task, there was a conflict with regular I/O refreshing, and overlapping refreshing occurred. ON if the I/O refreshing was completed normally.

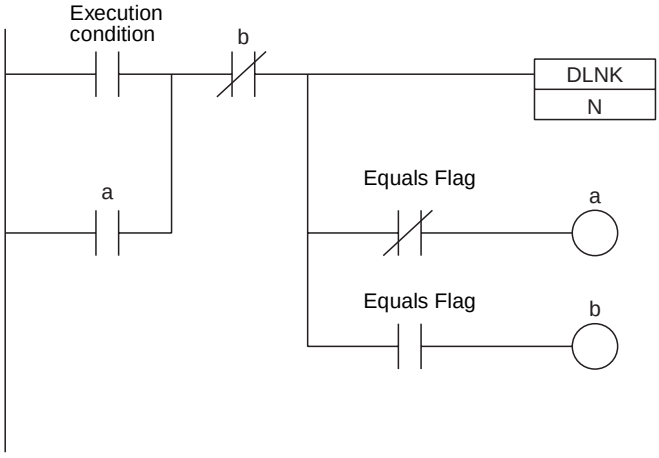
Precautions

I/O refreshing will not be performed if a CPU Bus Unit Error (A402.07) or CPU Bus Unit Setup Error (A402.03) has occurred in the specified CPU Bus Unit.

I/O refreshing will be stopped if an I/O Bus Error occurs while I/O refreshing is being performed by DLNK(226).

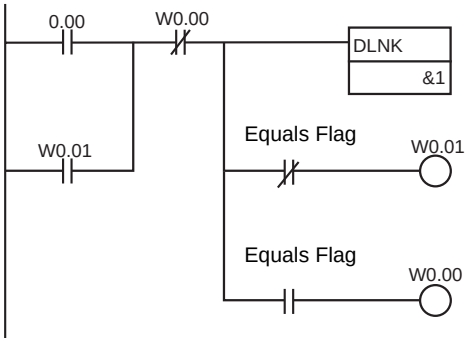
DLNK(226) refreshes data between the CPU Unit and specified CPU Bus Unit. Some time is required for the data exchange with the CPU Bus Unit (for example, a data link with a Controller Link Unit).

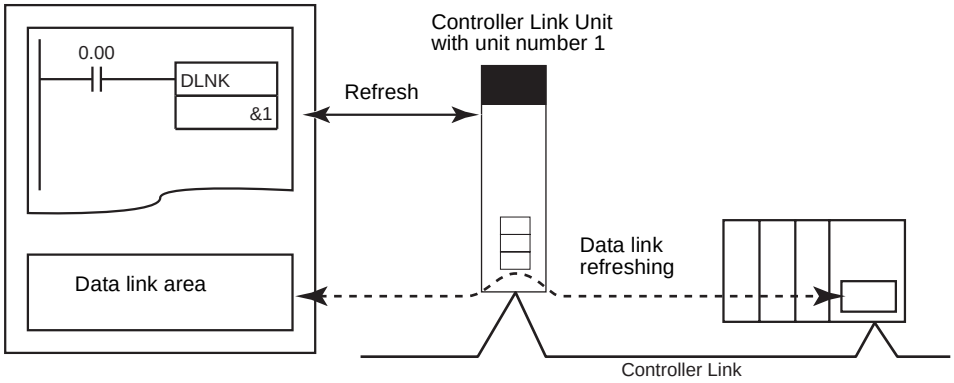
If the specified CPU Bus Unit is exchanging data, DLNK(226) will not be executed and the Equals Flag will be turned OFF. We recommend programming the execution conditions shown below so that the execution of DLNK(226) will be retried automatically.



Example

When CIO 0.00 is ON in the following example, DLNK(226) performs immediate I/O refreshing (in this case, data link refreshing within the PLC) for the CPU Bus Unit with unit number 1 (in this case, a Controller Link Unit).If I/O refreshing cannot be performed because the Controller Link Unit is refreshing data, the Equals Flag will be turned OFF causing W0.01 to be turned ON so that the instruction execution will be retried in the next cycle. When the I/O refreshing is completed normally, the Equals Flag will be turned ON and the instruction will not be retried in the next cycle.



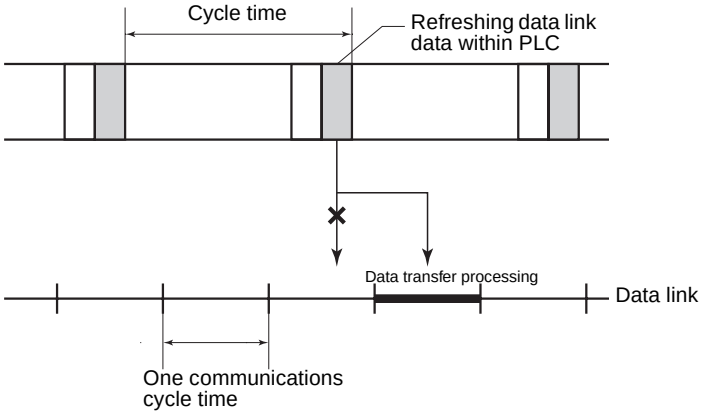


The actual timing for data link area refreshing in this example is as follows:

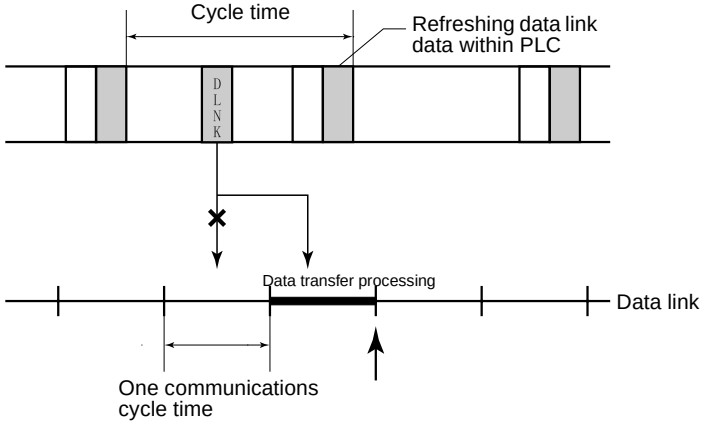
- When transmitting: Data is transmitted over the network the next time that the token right is acquired. (The transmitted data is delayed up to 1 communications cycle time max.)
- When receiving: The data that is input was received from the network the last time that the token right was acquired. (The data received is delayed up to 1 communications cycle time max.)

Examples of Data Transfer Processing:

- Transferring Data from the Previous I/O Refreshing



- Transferring Data with Execution of DLNK(226)



3-23 Serial Communications Instructions

This section describes instructions used for serial communications.

Instruction	Mnemonic	Function code	Page
PROTOCOL MACRO	PMCR	260	805
TRANSMIT	TXD	236	814
RECEIVE	RXD	235	819
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	256	824
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	255	832
CHANGE SERIAL PORT SETUP	STUP	237	840

3-23-1 Serial Communications

There are two types of serial communications instruction. The TXD(236), RXD(235), TXDU(256), and RXDU(255) instructions send and receive data in no-protocol (custom) communications with an external device. PMCR(260) sends and receives data using user-defined protocols with an external device. The difference is shown in the following tables.

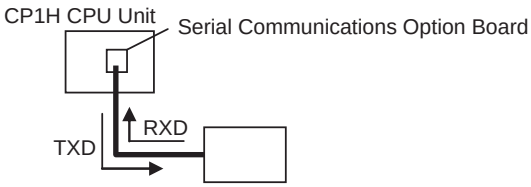
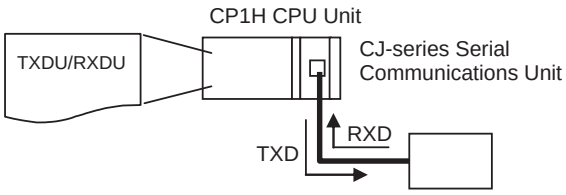
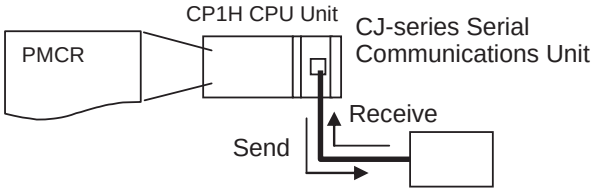
Note

(1) The TXD(236) and RXD(235) instructions transfer data only through a port on a Serial Communications Option Board.

(2) The TXDU(256) and RXDU(255) instructions transfer data only through a CJ-series Serial Communications Unit (Ver. 1.2 or later)

(3) The PMCR(260) instructions transfers data only through a CJ-series Serial Communications Unit.

Instructions	Communications frames	Function
TXD(236), RXD(235), TXDU(256), and RXDU(255)	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.
PMCR(260)	The following type of frames (messages) can be created to meet the requirements of the external device. Header | Address | Data | Error check | Terminator Communications steps can be created. I/O memory Read/write	Up to 16 steps can be defined for sending and receiving. Steps can be changed and retry processing performed based on responses. Communications monitoring times can be set. Symbols can be read/written for the PLC. Repeat symbols can be used. Other.

Instructions	Mode	Communications ports
TXD(236) and RXD(235)	No-protocol (custom)	Serial Port on a Serial Communications Option Board 
TXDU(256) and RXDU(255)	No-protocol (custom)	Serial Port of a CJ-series Serial Communications Unit (Version 1.2 or later) 
PMCR(260)	Protocol macro	Serial Port of a CJ-series Serial Communications Unit 

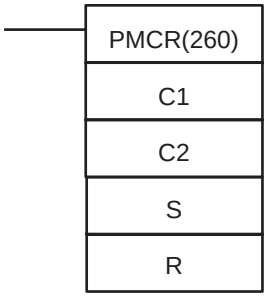
3-23-2 PROTOCOL MACRO: PMCR(260)

Purpose

Calls and executes a communications sequence registered in a CJ-series Serial Communications Unit.

Note A CJ Unit Adapter is required to use CJ-series Serial Communications Units.

Ladder Symbol



C1: Control word 1

C2: Control word 2

S: First send word

R: First receive word

Variations

Variations		
	Executed Each Cycle for ON Condition	PMCR(260)
	Executed Once for Upward Differentiation	@PMCR(260)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

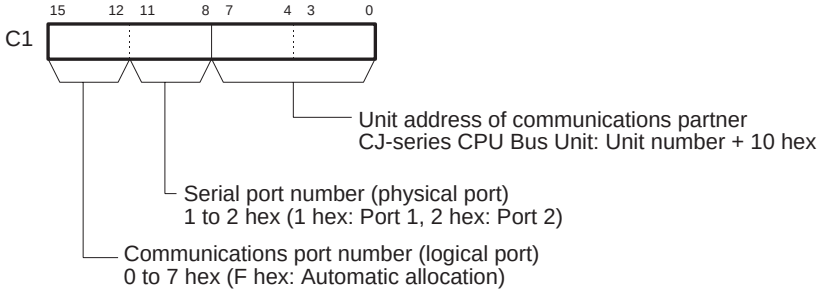
Applicable Program Areas

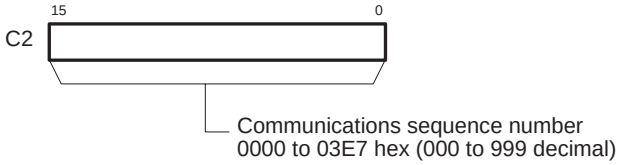
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C1: Control Word 1 and C2: Control Word 2

The contents of the two control words are shown below.

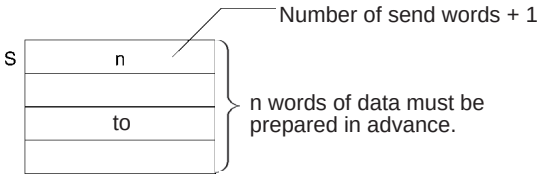




S: First Send Word and Send Area

The first word of the words required to send data is specified. S contains the number of words to be sent +1 (i.e., including the S word) and send data starts in S+1. Between 0000 and 00FA hex (0 and 250 decimal) words can be sent.

If there is no operand specified in the execution sequence, such as a direct or linked word, specify the constant #0000 for S. If a word address or register is specified, the data in the word or register must always be 0000. An error will occur and the Error Flag will turn ON if any other constant or a word address is given and PMCR(260) will not be executed.



R: First Receive Word and Receive Area

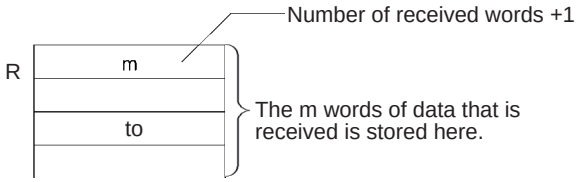
Received data is automatically stored in words starting with R+1 and the number of words received plus R (i.e., including R) is automatically written to R between 0000 and 00FA hex (0 and 250 decimal).

Setting Before Executing PMCR

Set the data specified by m (beginning with D) as the initial data for the receive buffer (backup data for receive failure). Data m can be set to 0002 to 00FA (hex) (2 to 255). If 0000 (hex) or 0001 (hex) is specified for m, the initial value of the receive buffer will be cleared to 0.

Always set a word address for R even if there is no receive data. If a constant is set, an error will occur, the Error Flag will turn ON, and PMCR(260) will not be executed. If there is no receive data, R will not be used and can be used for other purposes.

If there is no operand specified in the execution sequence, such as a direct or linked word, specify the constant #0000 for R. If a word address or register is specified, the data in the word or register must always be 0000.



Operand Specifications

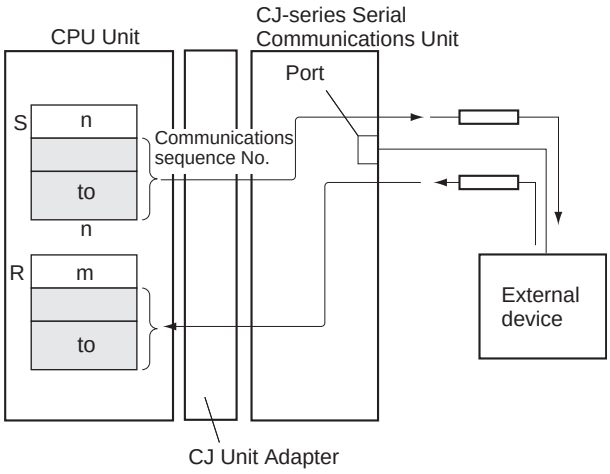
Area	C1	C2	S	R
CIO Area	CIO 0 to CIO 6143			
Work Area	W0 to W511			
Holding Bit Area	H0 to H511			
Auxiliary Bit Area	A0 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D0 to D32767			
Indirect DM addresses in binary	@ D0 to @ D32767			
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	Specified values only	0000 to 03E7 hex (0 to 999)	#0000 (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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

PMCR(260) will execute the communications sequence specified in C2 using the logical port specified in bits 12 to 15 of C1 and the physical port specified in bits 8 to 11 of C1 for the unit address specified in bits 0 to 7 of C1.

If a symbol is specified as the operand for a send message, the number of send words specified in S and beginning from S+1 will be used as the send area. If a symbol is specified as the operand for a receive message, receive data is placed in memory staring with R+1 and the number of words received is automatically written to R if the transmission is successful.

If the transmission fails, the data (R+1 onward) set before PMCR(260) was executed will be read from the receive buffer and stored in the R+1 onward again.



Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the specified logical port when PMCR(260) is executed. ON if C1 is not within the specified ranges. (Error flag will not turn ON if the C2 data is outside the specified ranges. The end code will be stored in the Communications Port Completion Code (A203 to A210) of the auxiliary area.) ON if the number of words of S or R exceeds 249 (when words are specified). OFF in all other cases.

Precautions

The data in the send area specified with S is actually sent using the symbol read option, R( ), in a send message.

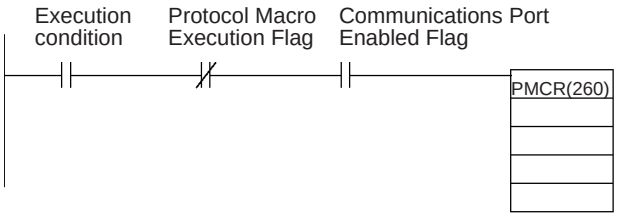
Data is actually received to the receive area specified by R using the symbol write option, W( ), in a receive message.

Refer to the *CX-Protocol Operation Manual (W344)* for procedures for designating symbols R( ) and W( ).

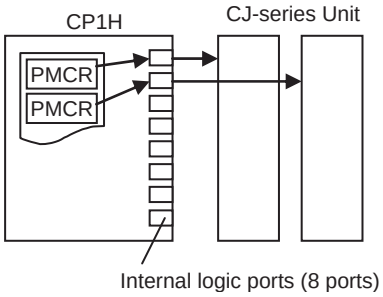
PMCR(260) can be executed for a serial communications port on a Serial Communications Unit. Up to 2 Serial Communications Units can be mounted as CJ-series Expansion Units. The unit address of the communications partner must be set in bits 0 to 7 of C1 to specify which Unit is to be used and the serial port number must be set in bits 8 to 11. Unit addresses are specified as shown in the following table.

Unit/Board	Unit address
Serial Communications Unit	Unit number + 10 hex

The corresponding Protocol Macro Execution Flag will turn ON at the start of PMCR(260) execution. It will turn OFF after the communications sequence has been completed and data has been written to the specified receive area. A N.C. input for the corresponding Protocol Macro Execution Flag should be used as part of the execution condition whenever executing PMCR(260) to be sure that only one communications sequence is being executed at the same time for the same physical port. An example is shown below.



SEND(090), RECV(098), and CMND(490) also use the logical ports 0 to 7 to execution communications sequences through Serial Communications Unit (internally using FINS commands). PMCR(260) cannot be executed for a logical port that is already being used by SEND(090), RECV(098), CMND(490)or PMCR(260). To prevent more than one communications sequence from being executed for the same logical port, the corresponding Communications Port Enable Flag (A202.00 to A202.07) should be used as a N.O. input in the execution condition for PMCR(260), as shown in the above diagram.



The Error Flag will turn ON in the following cases.

- The corresponding Communications Port Enable Flag is OFF for the specified logical port (0 to 7) when PMCR(260) is executed.
- C1 is not within the specified ranges.

Designation of Receive Area

Before executing PMCR(260), users must set backup data in the receive area for receive processing failure. Once the PMCR(260) is executed, the data in the receive buffer is automatically stored in the receive area. One example of the backup data application is as follows: A certain value (backup data) is set in advance so that the present value will not be read as zero when transmission failure occurs while protocol is being executed for reading the present value of a controller.

Related Flags and Words

The following flags and words can be used as required when executing PMCR(260).

Auxiliary Area

Name	Address	Contents
Communications Port Enabled Flag	A202.00 to A202.07	ON when network communications are enabled (including PMCR(260). Bits 00 to 07 correspond to logical ports 0 to 7, respectively. A Communications Port Enabled Flag will turn OFF when network communications are started and will turn ON when they are completed (regardless of whether communications end normally or in error.
Communications Port Error Flag	A219.00 to A219.07	ON when an error occurs in network communications. Bits 00 to 07 correspond to logical ports 0 to 7, respectively. Flag status will be maintained until the next network communications start. The flag will turn OFF when communications start again even if an error occurred for the last execution.
Communications Port Completion Codes	A203 to A210	Contains the completion code stored when network communications are performed. Words A203 to A210 correspond to logical ports 0 to 7, respectively. The completion code will be 00 while the communications instruction is being executed. The new response code will be stored when execution has been completed. The contents of these words is cleared when operation is started.

Communications Responses

Code	Contents
1106 (hex)	No corresponding program number Specified Send/Receive Sequence No. that has not been registered. Modify the Send/Receive Sequence No. or add the number using the CX-Programmer.
2201 (hex)	Not operable due to protocol execution Since one protocol macro has already been executed, no further execution is accepted. Add NC condition to program for the Protocol Macro Execution Flag.
2202 (hex)	Not operable due to stoppage Since the protocol is being switched, no further execution is accepted. Add NC condition to program for the Serial Setting Change Flag.
2401 (hex)	No registration table An error has occurred in the protocol macro data or data is being transmitted. Transmit the protocol macro data using the CX-Programmer.
Others	Refer to the *CJ-series Communications Commands Reference Manual (W342)* for other response codes.

CPU Bus Unit Area

n = 1500 + 25 x unit number

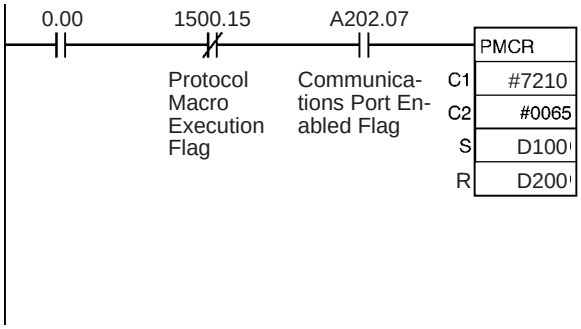
Name	Address	Contents
Port 1 Protocol Macro Execution Flag	Bit 15 of CIO n+9	ON when PMCR(260) is executed. The flag will remain OFF if execution fails. The flag will turn OFF when the communications sequence has been completed (either an end or abort).
Port 2 Protocol Macro Execution Flag	Bit 15 of CIO n+19	

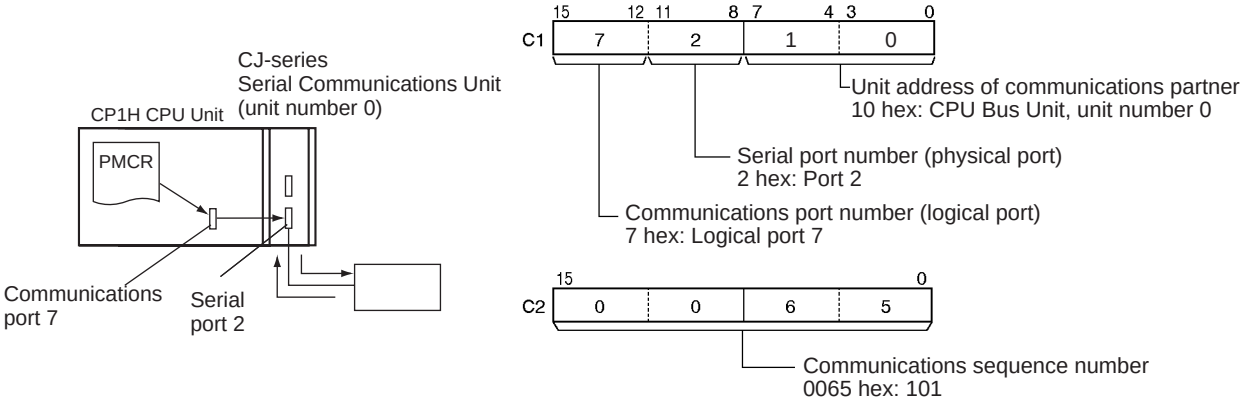
Examples

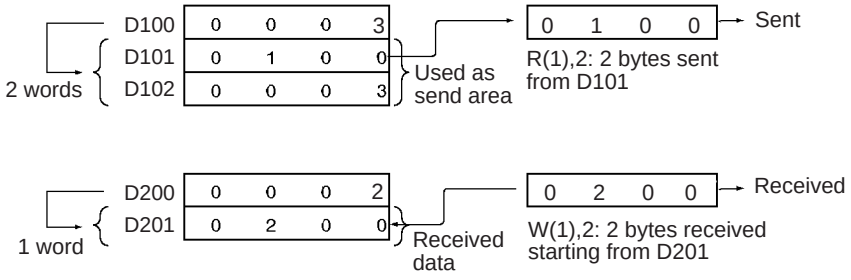
When CIO 0.00 is ON in the following example, communications sequence No. 101 (0065 hex) will be executed for port 2 on the Serial Communications Unit with unit number 0 as long as the Communications Port Enabled Flag for port 7 (A202.07) is ON and the Protocol Macro Execution Flag (CIO 1500.15) is OFF.

If an operand is specified for the symbol in a send message, 2 words of data starting from D101 will be used as the send area (because the contents of D100 is #0003).

If an operand is specified for the symbol in a receive message, 2 words of data will be stored starting from D201 and the number of words received +1 will be written to D200.







Note As shown above, the symbol read option, R( ), in the send message or the symbol write option, W( ), actually sends/receives data.

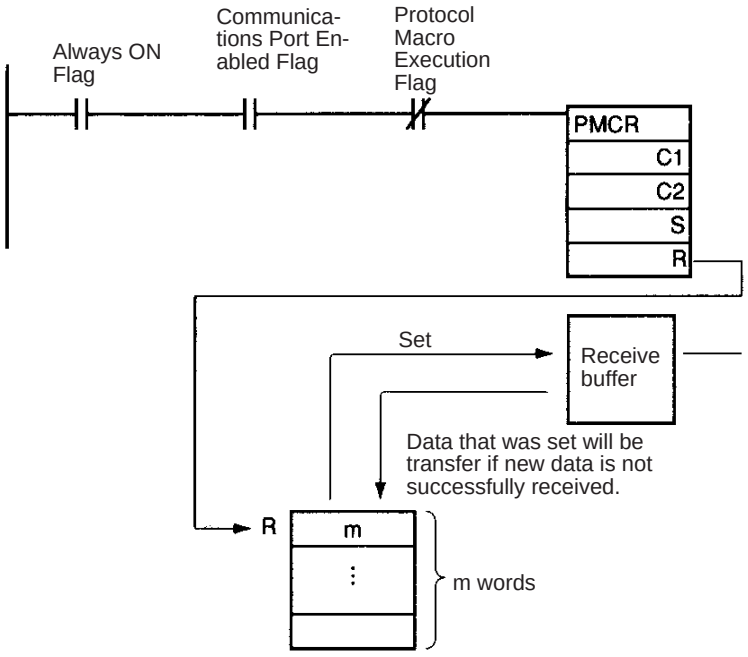
Holding the Receive Area

The receive buffer is cleared to all zeros immediately before a communications sequence is executed for PMCR(260). If programming such as that shown below is used to periodically read PV data or other values and data cannot be read due to a reception error or other cause, the data being read will be cleared until the next successful read.

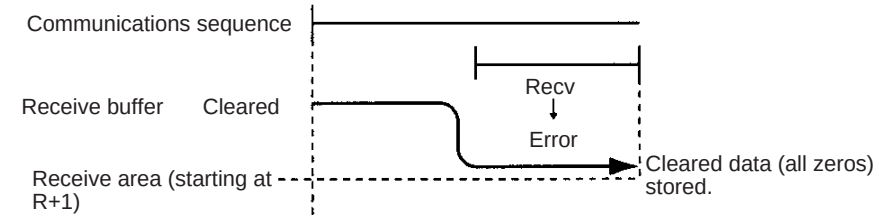
A function is provided to maintain the data in the receive area even when a reception error occurs. If this function is used, data will be transferred from the first m words of the receive area to the receive buffer after the buffer is cleared to all zeros but before the communications sequence is executed. This prevents the receive area from being temporarily cleared to all zeros by writing the most recent receive data when new receive data is not successfully obtained.

Specify the number of words of the receive area to be maintained as the value m. If 0 or 1 is specified, the holding function will be disabled and the receive area will be cleared to all zeros.

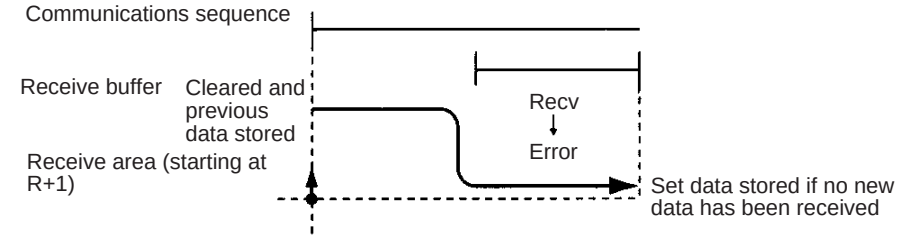
The following programming example shows the instructions used to constantly or periodically execute PMCR(260) to read data through a single receive operation.



Receive Area Not Held



Receive Area Held

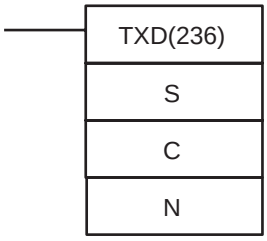


3-23-3 TRANSMIT: TXD(236)

Purpose

Outputs the specified number of bytes of data from a serial port on a Serial Communications Option Board mounted to the CPU Unit.

Ladder Symbol



S: First source word

C: Control word

N: Number of bytes
0000 to 0100 hex (0 to 256)

Variations

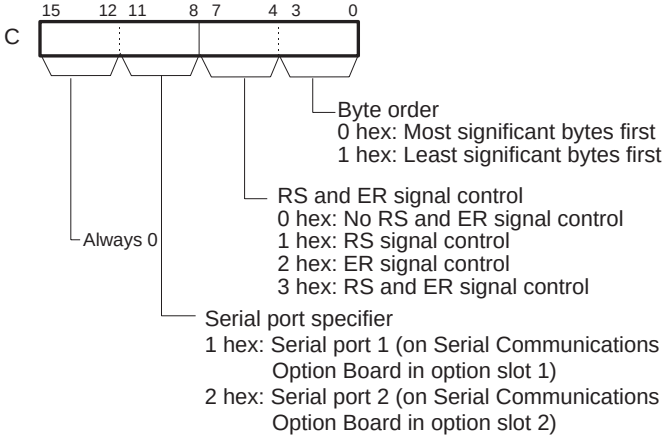
Variations	**Executed Each Cycle for ON Condition**	TXD(236)
	Executed Once for Upward Differentiation	@TXD(236)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A447 A448 to A959		
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		

Area	S	C	N
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
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

TXD(236) reads N bytes of data from words S to S+(N÷2)–1 and outputs the raw data in no-protocol mode (RS-232C mode) from a serial port on a Serial Communications Option Board mounted to the CPU Unit. (The output port is specified with bits 8 to 11 of C.)

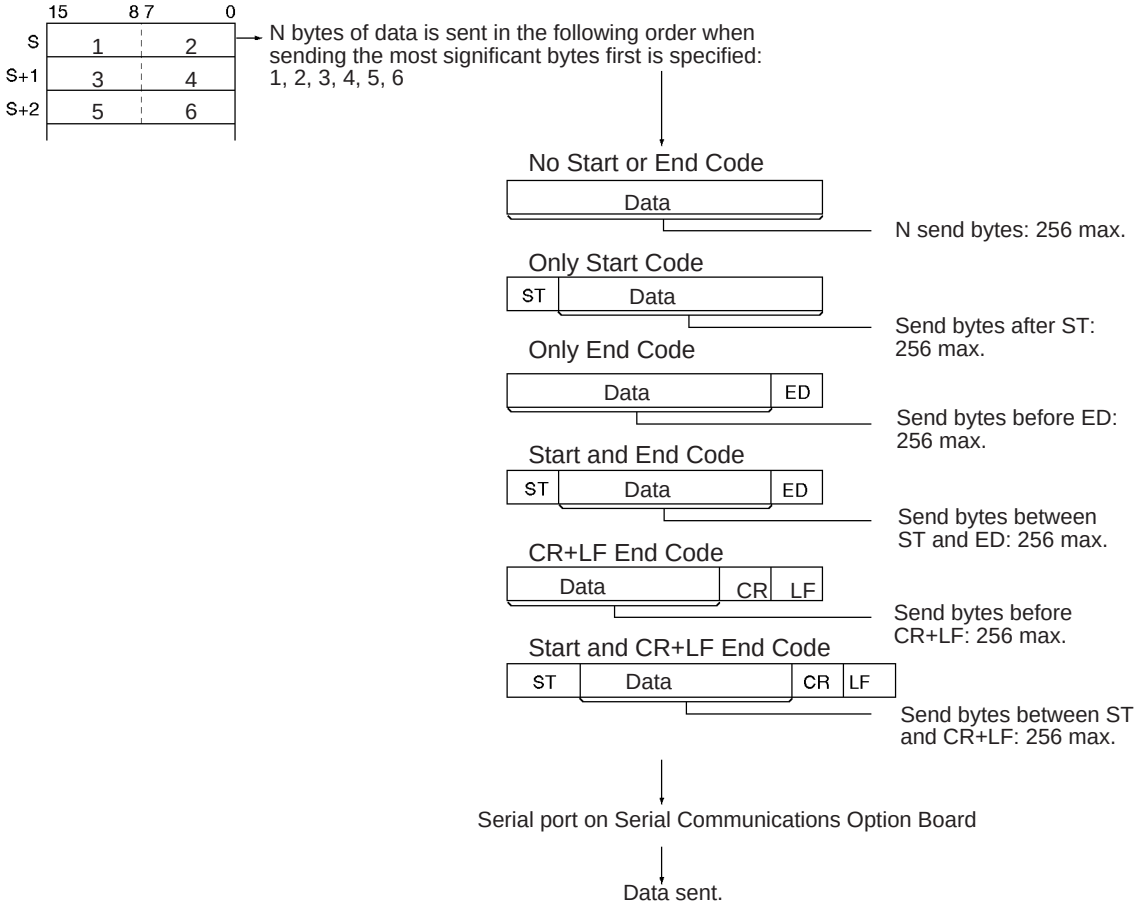
The start and end codes specified for no-protocol mode are added to the data before the data is output. The start and end codes are specified in the PLC Setup.

Data can be sent only when the port's Send Ready Flag is ON. The Send Ready Flag is A392.13 serial port 1 and A392.05 for serial port 2.

Up to 259 bytes can be sent, including the send data (N = 256 bytes max.), the start code, and the end code.

Note Serial port 1 is the port on the Serial Communications Option Board mounted in option slot 1 and serial port 2 is the port on the Serial Communications Option Board mounted in option slot 2.

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 no-protocol mode (RS-232C mode) is not set in the PLC 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 is OFF. OFF in all other cases.

Precautions

TXD(236) can be used only for Serial Communications Option Board's serial ports that are set to no-protocol mode (RS-232C mode).

The following send-message frame format can be set in the PLC 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 PLC Setup. If start and end codes are specified, the codes will be added to the send data (N). In this case, the maximum number of bytes that can be specified for N is 256 bytes.

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.

An error will occur and the Error Flag will turn ON in the following cases.

- No-protocol mode is not set for the port in the PLC Setup.
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.
- A send was attempted when the Send Ready Flag was OFF.

Related PLC Setup Settings

The following PLC Setup settings must be made for the serial port 1 or serial port 2 executing TXD(236) after setting the communications mode to no-protocol (RS-232C mode).

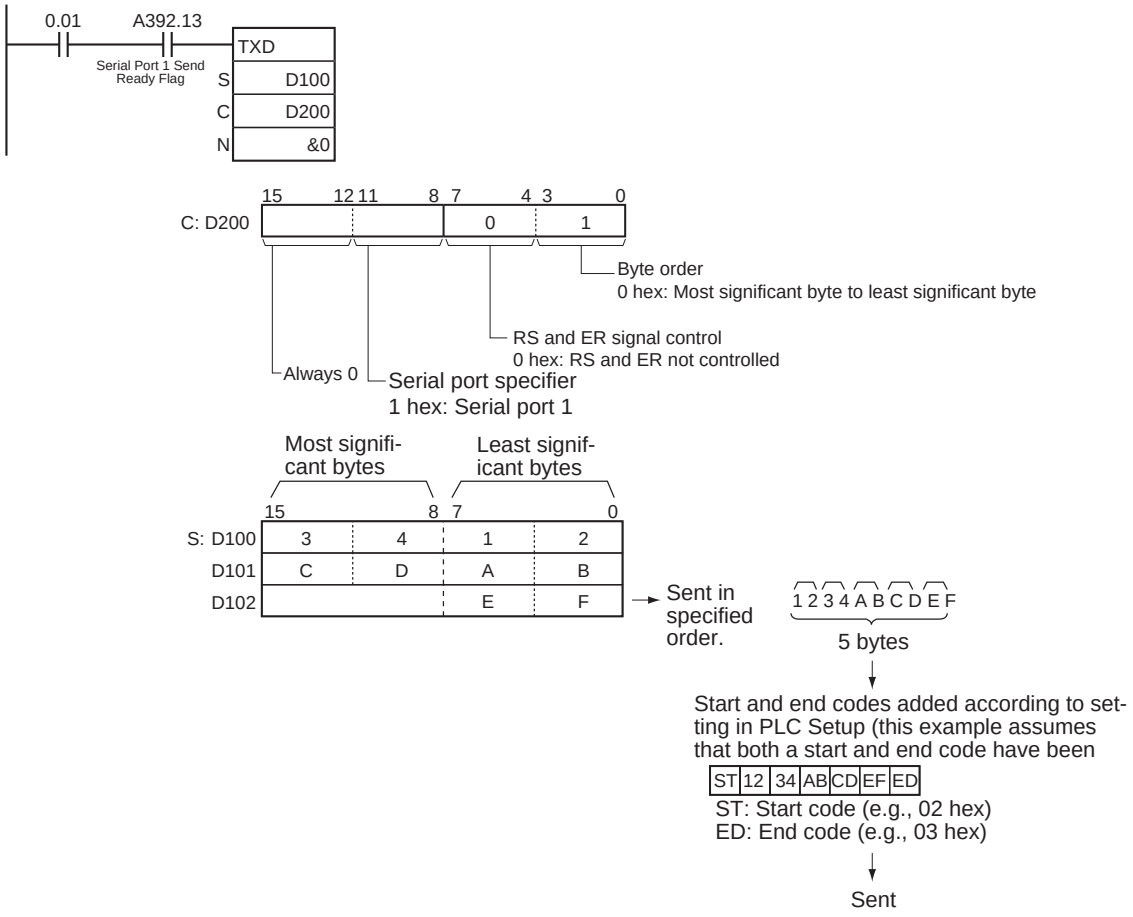
Setting		Description	
Communications Settings		Standard	The standard settings are as follows: 9,600 baud, 1 start bit, 7-bit data, even parity, and 2 stop bits.
		Custom	Baud: 300, 600, 1,200, 2,400, 4,800, 9,600, 19,200, 38,400, 57,600, or 115,200 bps Format: 7- or 8-bit data; 1 or 2 stop bits; even, odd, or even parity
Start Code		Disable	---
		Set	00 to FF hex
End Code	Received Bytes	No	---
		Yes	00 hex: 256 bytes 01 to FF hex: 1 to 255 bytes
	End Code	No	---
		CR/LF	---
		Set	00 to FF hex
Delay		0000 to 210F hex: 0 to 99,990 ms (10-ms units)	

Auxiliary Area

Name	Address	Contents
Serial Port 1 Send Ready Flag	A392.13	ON when data can be sent in no-protocol mode.
Serial Port 2 Send Ready Flag	A392.05	

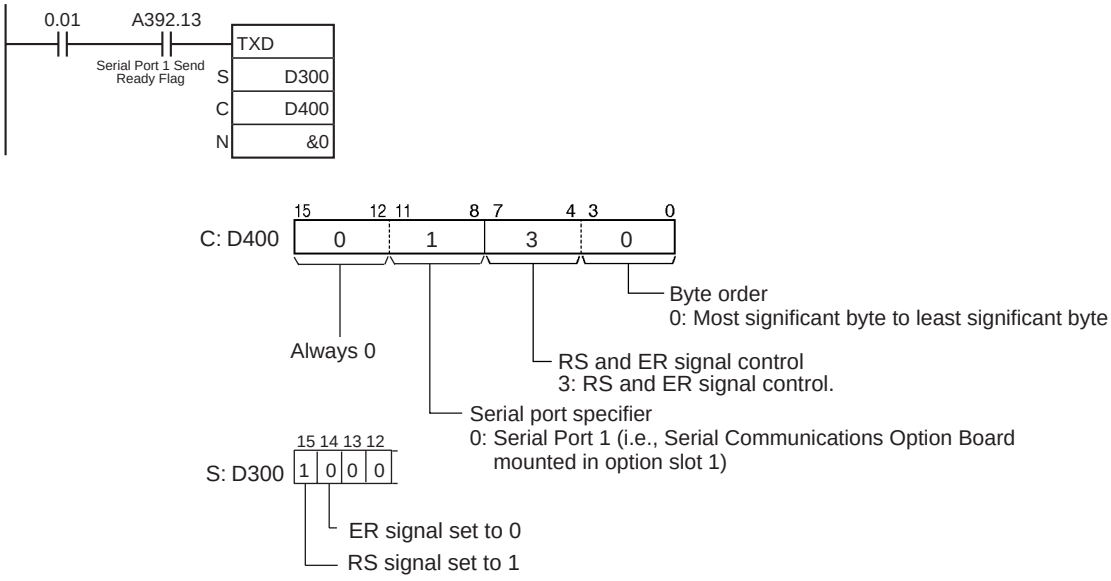
Example: Sending Data

When CIO 0.01 and the Serial Port 1 Send Ready Flag (A392.13) are ON in the following example, five bytes of data starting from the lower byte of D100 is sent to the Serial Communications Option Board mounted in option slot 1.



Example: Performing Signal Control

When CIO 0.01 and the Serial Port 1 Send Ready Flag (A392.13) are ON in the following example, the RS signal is set according to the status of D300 bit 15 and the ER signal is set according to the status of D300 bit 14.

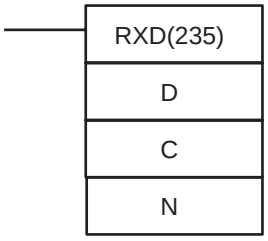


3-23-4 RECEIVE: RXD(235)

Purpose

Reads the specified number of bytes of data from a serial port on a Serial Communications Option Board mounted to the CPU Unit.

Ladder Symbol



D: First destination word

C: Control word

N: Number of bytes to store
0000 to 0100 hex (0 to 256 decimal)

Variations

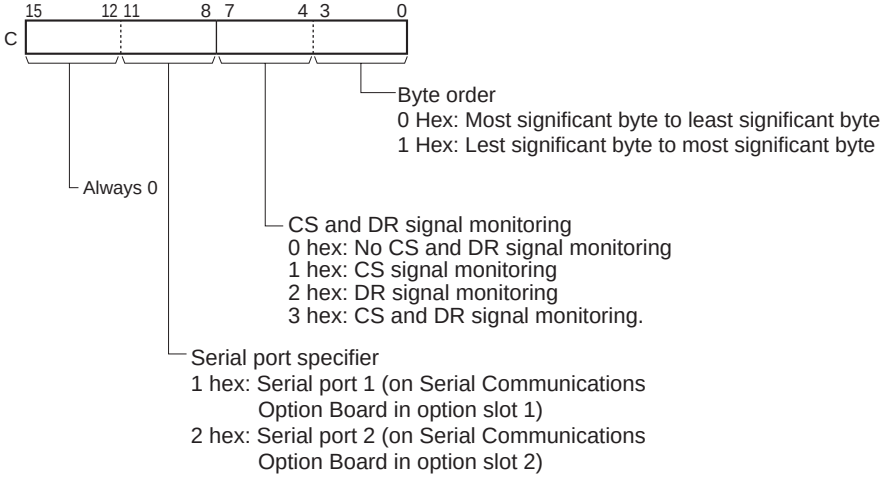
Variations	**Executed Each Cycle for ON Condition**	RXD(235)
	Executed Once for Upward Differentiation	@RXD(235)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A448 to A959	A0 to A447 A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		

Area	D	C	N
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
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

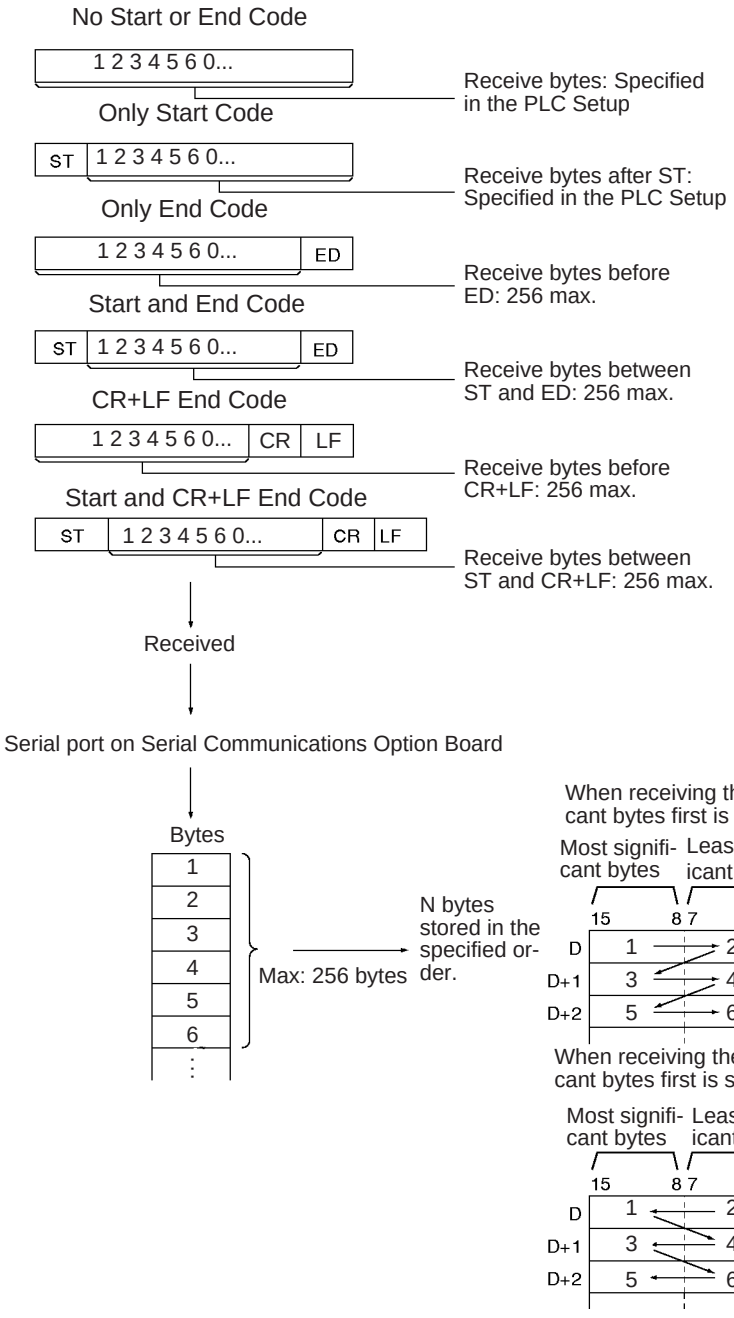
RXD(235) reads data that has been received in no-protocol mode at a serial port on a Serial Communications Option Board and stores N bytes of data in words D to D+(N÷2)–1. If N bytes of data has not been received at the port, then only the data that has been received will be stored.

Data can be received only when the port's Receive Ready Flag is ON. The Receive Ready Flag is A392.14 for serial port 1 and A396.06 for serial port 2. Execute RXD(235) only when the corresponding Receive Ready Flag is ON.

Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.

The following diagram shows the order in which data is received and the contents of the receive frame for various settings.

Note Serial port 1 is the port on the Serial Communications Option Board mounted in option slot 1 and serial port 2 is the port on the Serial Communications Option Board mounted in option slot 2.



Flags

Name	Label	Operation
Error Flag	ER	ON if no-protocol mode is not set in the PLC 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 serial ports on Serial Communications Option Boards mounted to the CPU Unit. In addition, the port must be set to no-protocol mode.

The following receive message frame format can be set in the PLC 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 received is set from 00 to FF hex (1 to 256 decimal; 00 specifies 256 bytes).

The Reception Completed Flag will turn ON when the number of bytes specified in the PLC Setup has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter will have the same value as the number of receive bytes specified in the PLC Setup or the allocated DM Setup Area. If more bytes are received than specified, the Reception Overflow Flag will turn ON.

If an end code is specified in the PLC Setup, the Reception Completed Flag 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 turns ON, the Reception Overflow Flag will turn ON.

When RXD(235) is executed, data is stored in memory starting at D, the Reception Completed Flag will turn OFF, and the Reception Counter will be cleared to 0. If the Reception Overflow Flag has turned ON, it will also turn OFF.

Data will be stored in memory in the order specified in C.

If 0 is specified for N, the Reception Completed Flag will be turned OFF, the Reception Counter 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 RS and ER signal control in C, RXD(235) will be executed regardless of the status of the Receive Completed Flag.

An error will occur and the Error Flag will turn ON in the following cases.

- The no-protocol mode (RS-232C mode) is not set for the port in the PLC Setup.
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.

Read the data using RXD(235) as soon as possible after the Reception Completed Flag turns ON. If reception is continued without reading the data, the reception buffer will overflow and data reception will stop. If this occurs, the port will have to be restarted to recover operation.

The reception buffer is not cleared when RXD(235) is executed. Thus, more than one RXD(235) instruction can be used to read the data.

Related PLC Setup Settings

The following PLC Setup settings must be made for the serial port 1 or serial port 2 executing RXD(235). after setting the communications mode to no-protocol (RS-232C mode).

Setting		Description	
Communications Settings		Standard	The standard settings are as follows: 9,600 baud, 1 start bit, 7-bit data, even parity, and 2 stop bits.
		Custom	Baud: 300, 600, 1,200, 2,400, 4,800, 9,600, 19,200, 38,400, 57,600, or 115,200 bps Format: 7- or 8-bit data; 1 or 2 stop bits; even, odd, or even parity
Start Code		Disable	---
		Set	00 to FF hex
End Code	Received Bytes	No	---
		Yes	00 hex: 256 bytes 01 to FF hex: 1 to 255 bytes
	End Code	No	---
		CR/LF	---
		Set	00 to FF hex
Delay		0000 to 210F hex: 0 to 99,990 ms (10-ms units)	

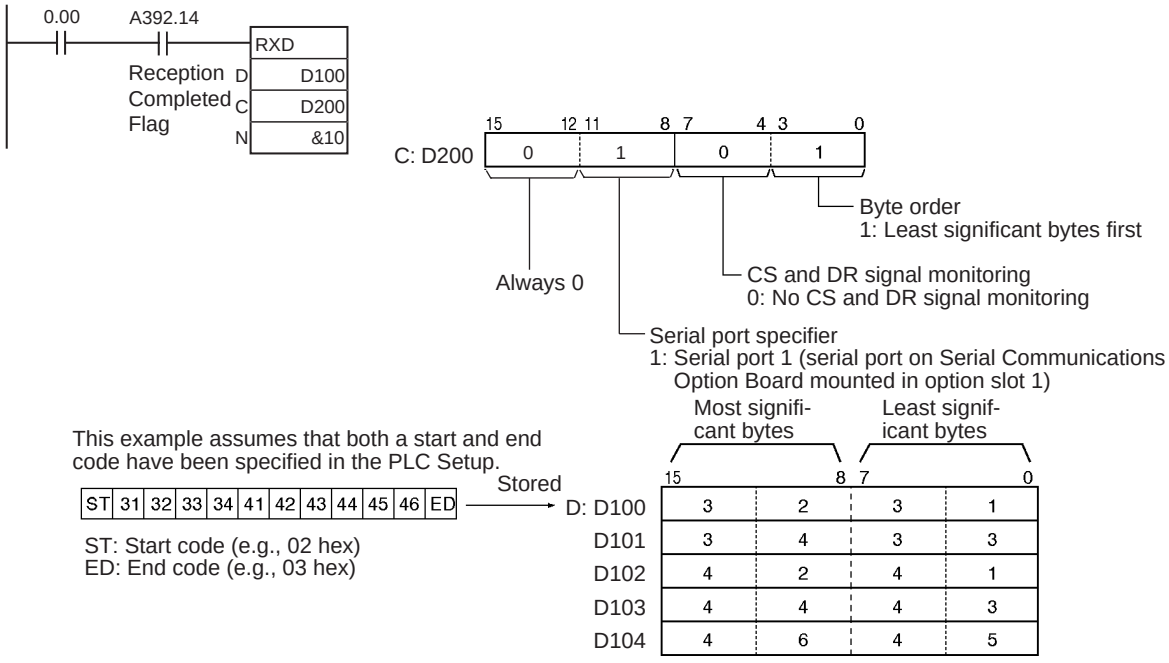
Auxiliary Area Flags

Name	Address	Contents
Serial Port 1 Reception Completed Flag	A392.14	ON when the serial port has completed the reception in no-protocol mode. When the number of bytes was specified: ON when the specified number of bytes is received. When the end code was specified: ON when the end code is received or 256 bytes are received.
Serial Port 2 Reception Completed Flag	A392.06	
Serial Port 1 Reception Overflow Flag	A392.15	ON when a data overflow occurred during reception through a serial port in no-protocol mode. .When the number of bytes was specified: ON when more data is received after the reception was completed but before RXD(235) was executed. When the end code was specified: ON when more data is received after the end code was received but before RXD(235) was executed. ON when 257 bytes are received before the end code.
Serial Port 2 Reception Overflow Flag	A392.07	
Serial Port 1 Reception Counter	A394	Indicates (in hexadecimal) the number of bytes of data received when the serial port is in no-protocol mode.
Serial Port 2 Reception Counter	A393	

Name	Address	Contents
Serial Port 1 Restart Bit	A526.01	Turn this bit ON to restart the serial port. The Reception Completed Flag and Reception Overflow Flag will be turned OFF and the Reception Counter will be cleared to 0. This bit is turned OFF automatically when the restart processing is completed.
Serial Port 2 Restart Bit	A526.00	

Examples

When CIO 0.00 and the Serial Port 1 Reception Completed Flag (A392.14) is ON in the following example, data is received from serial port 1 and 10 bytes of data are stored starting in D100.



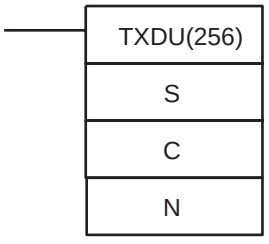
3-23-5 TRANSMIT VIA SERIAL COMMUNICATIONS UNIT: TXDU(256)

Purpose

Outputs the specified number of bytes of data from one of the CJ-series Serial Communications Unit's serial ports.

Note A CJ Unit Adapter is required to use CJ-series Serial Communications Units.

Ladder Symbol



S: First source word

C: First control word

N: Number of bytes
0000 to 0100 hex (0 to 256)

Variations

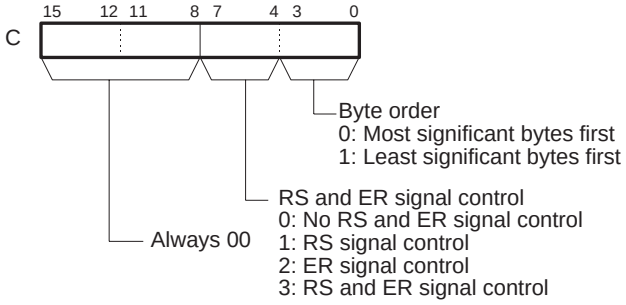
Variations		
	Executed Each Cycle for ON Condition	TXDU(256)
	Executed Once for Upward Differentiation	@TXDU(256)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

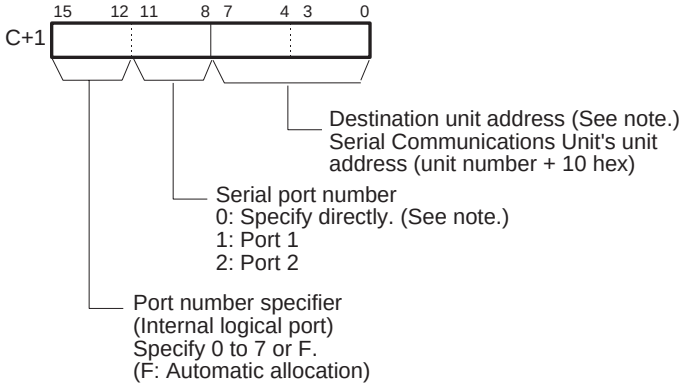
Applicable Program Areas

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

Operands

The contents of the control words, C and C+1, are as shown below.





Note The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the destination unit address to the serial port's unit address. (Set the destination unit address to 80 hex + 4 × unit number for port 1 or 81 hex + 4 × unit number for port 2.)

Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A958	A0 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
Data Registers	---	---	DR0 to DR15

Area	S	C	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

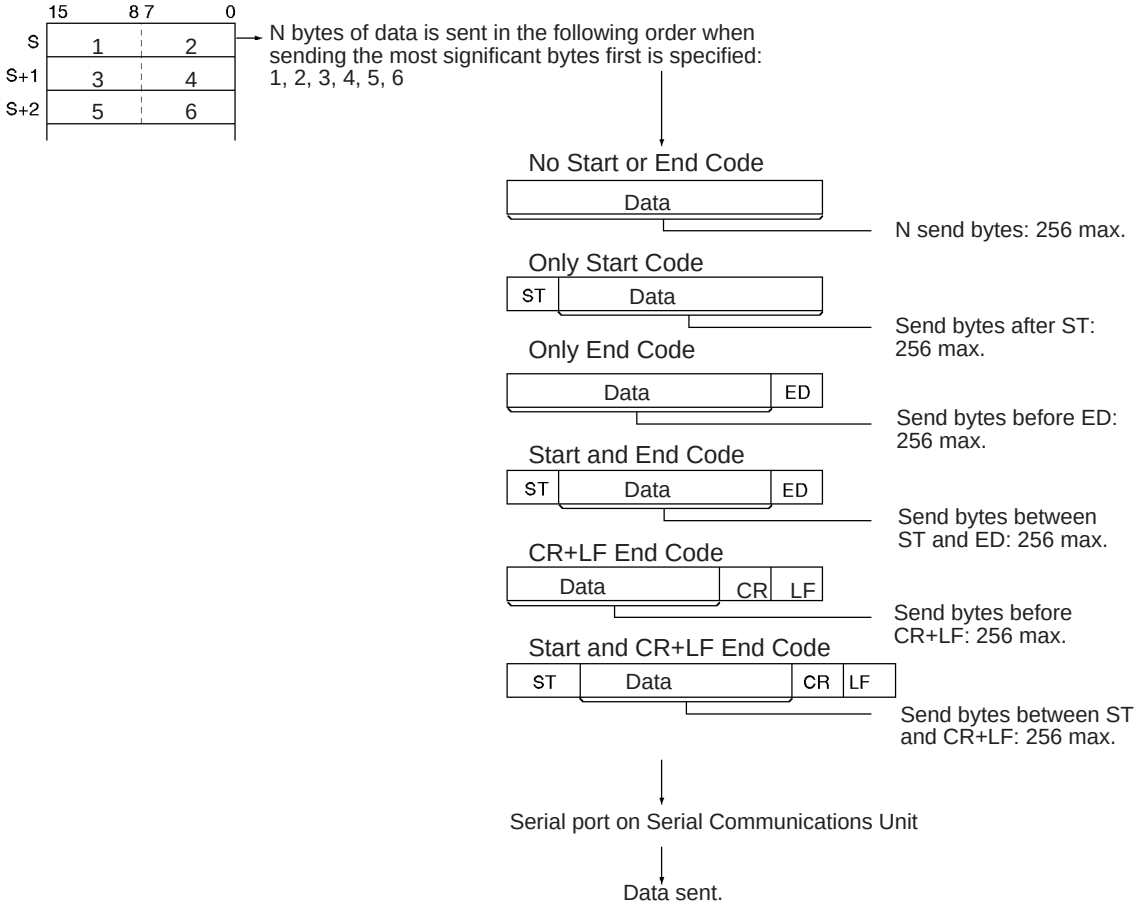
TXDU(256) reads N bytes of data from words S to S+(N÷2)–1 and outputs the raw data in no-protocol mode from the Serial Communications Unit with the unit address specified in bits 0 to 7 of C+1, through the port specified with bits 8 to 11 of C+1. The logical port number can be set to any value between 0 and 7 and is specified with bits 12 to 15 of C+1.

The start and end codes specified for no-protocol mode in the allocated DM Setup Area are added to the data before the data is output. Up to 259 bytes can be sent, including the send data (N = 256 bytes max.), the start code, and the end code.

Data can be sent only when the Communications Port Enabled Flag for the specified logical port (A202.00 to A202.07 for ports 0 to 7) is ON and the TXDU Instruction Executing Flag (in the allocated DM Setup Area) is OFF.

Note The logical port number can be allocated automatically by setting bits 12 to 15 of C+1 to F. For details, refer to *About Communications Port Numbers* on page 849.

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 all of the logical ports are being used or the Communications Port Enabled Flag for the specified logical port is OFF when the instruction is executed. ON if the value of C is not within range. ON if the value for N is not between 0000 and 0200 hex. OFF in all other cases.

Precautions

TXDU(256) can be used only for a Serial Communications Unit's serial port that has been set to no-protocol mode.

The following send-message frame formats can be set in the allocated DM Setup Area.

- 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 combination of start and/or end codes specified in the allocated DM Setup Area. If start and end codes are specified, the codes will be added to the send data (N). 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 is ON.

Data is sent in the order specified in C.

Nothing will be sent if 0 is specified for N.

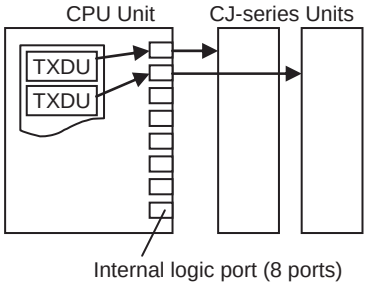
If RS signal control is specified in C, bit 15 of S will be used as the RS signal.

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.

TXDU(256) uses a logical port (because it sends an internal FINS command) to output a send sequence command to the Serial Communications Unit (version number 1.2 or later). Since SEND(090), RECV(098), CMND(490), PMCR(260), and RXDU(255) also use logical ports 0 to 7, TXDU(256) cannot be executed for a logical port if that logical port is already being used by one of those instructions or another TXDU(256) instruction.

To ensure that TXDU(256) is not executed while the logical port is busy, program the port's Communications Port Enabled Flag (A202.00 to A202.07) as a normally open condition.



TXDU(256) can not be executed while the TXDU Instruction Executing Flag (bit 5 of n+9 or n+19, where n = CIO 1500 + 25 × unit number) is ON. To ensure that another TXDU(256) is not executed for the port before the first TXDU(256) is completed, program the port's TXDU Instruction Executing Flag as a normally closed condition.

An error will occur and the Error Flag will turn ON in the following cases.

- The Communications Port Enabled Flag for the specified logical port is OFF when TXDU(256) is executed.
- The value of C is not within range.
- The value for N is not between 0000 and 0200 hex.

Note Depending on the external device, it might be necessary to set a send delay when sending data with TXDU(256). It a send delay is required, set or adjust the delay time in the allocated DM Setup Area.

Related Flags and Words

The following PLC Setup settings and Auxiliary Area flag can be used as required when executing TXD(236).

DM Setup Area Settings

(m = D30000 + 100 × unit number)

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
m+2	m+12	15	No-protocol Mode Send Delay Specifier	0: Default (0 ms) 1: Use delay in bits 1 to 14.
		0 to 14	No-protocol Mode Send Delay Time	0000 to 7530 hex 0 to 300,000 ms decimal (in 10-ms units)
m+4	m+14	8 to 15	No-protocol Mode Start Code	00 to FF hex
		0 to 7	No-protocol Mode End Code	00 to FF hex

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
m+5	m+15	12 to 15	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0: None 1: Use end code. 2: Use CR+LF.

Auxiliary Area

Name	Address	Description
Communications Port Enabled Flags	A202.00 to A202.07	ON when a communications instruction (including TXDU(256) can be executed with the corresponding port number. Bits 00 to 07 correspond to communications ports 0 to 7. The flag is OFF when a communications instruction is being executed and ON when the execution is completed (normal end or error end).
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding port numbers when communications instructions have been executed. Words A203 to A210 correspond to communications ports 0 to 7. The code is 00 while the instruction is being executed and contains the relevant code when execution is completed. These words are cleared to 0000 when PLC operation starts.
Communications Port Error Flags	A219.00 to A219.07	ON when an error occurred during execution of a communications instruction. When a flag is ON, check the completion code in A203 to A210 to troubleshoot the error. OFF when execution has been finished normally. Bits 00 to 07 correspond to communications ports 0 to 7. The flag status is retained until the next communications instruction is executed. Even if an error has occurred, a flag will be reset to 0 the next time that a communications instruction is executed for that port.

Completion Codes

Code	Meaning
0205 hex	Response timeout (This error can occur when the communications mode is set to host link mode.)
0401 hex	Undefined command (This error can occur when the communications mode is set to protocol macro, NT Link, echoback test, or serial gateway mode.)
1001 hex	The command is too long.
1002 hex	The command is too short.
1003 hex	The specified number of data elements does not match the actual amount of send data.
1004 hex	The command format is incorrect.
110C hex	Other parameter error
2201 hex	Operation could not be performed during operation. (Operation disabled because Unit is busy sending.)
2202 hex	Operation could not be performed when stopped. (Operation disabled because Unit is switching protocols.)

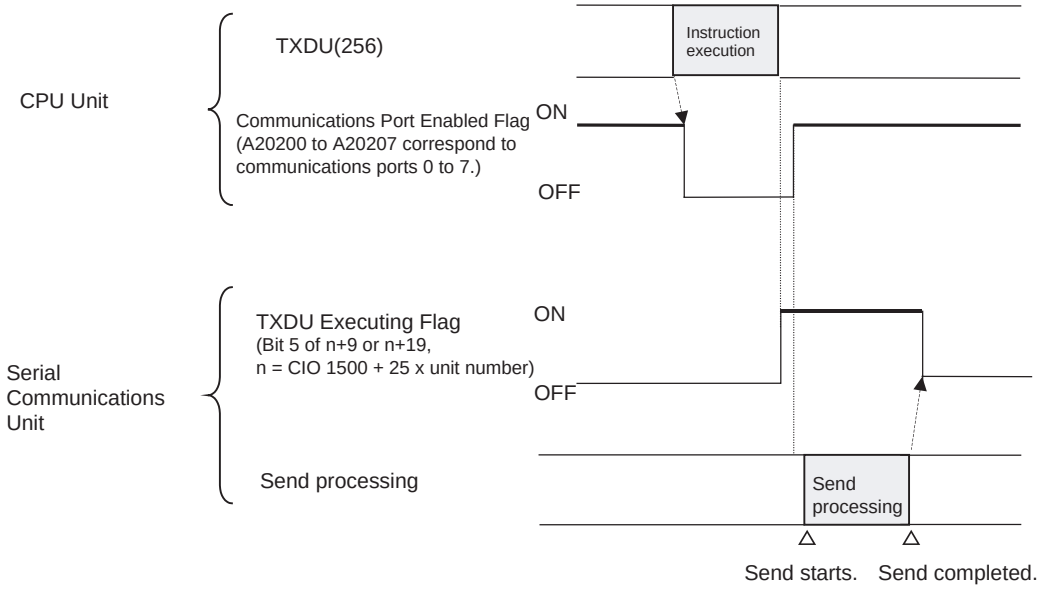
Related Flags in the CPU Bus Unit Area

(n = CIO 1500 + 25 × unit number)

Word		Bit	Name	Status
Port 1	Port 2			
n+9	n+19	05	TXDU Instruction Executing Flag	0: TXDU(256) is not being executed. 1: TXDU(256) is being executed.

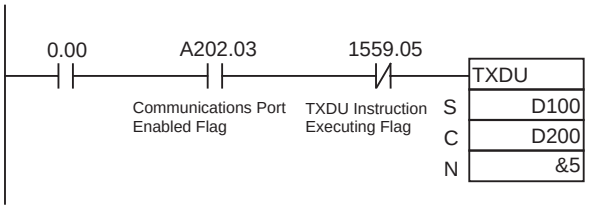
Example: Flag Operation

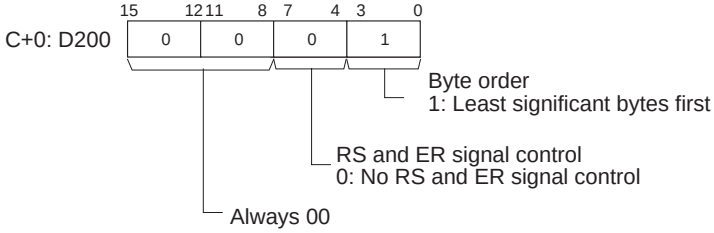
The following diagram shows the operation of the Communications Port Enabled Flag and TXDU Instruction Executing Flag.

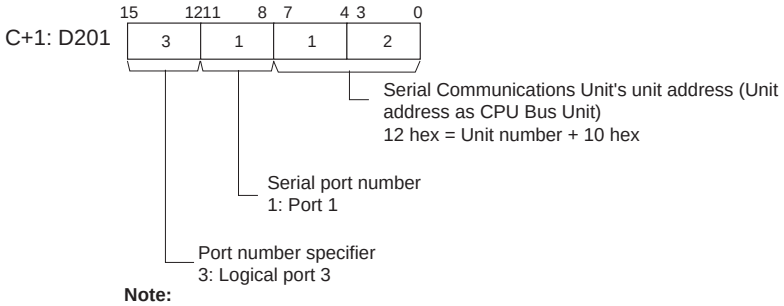


Example: Sending Data

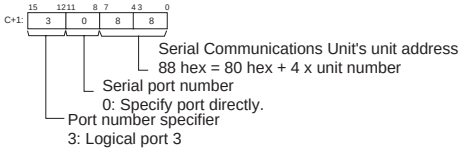
When CIO 0.00 is ON, A202.03 (the Communications Port Enabled Flag) is ON, and CIO 1559.05 (the TXDU Instruction Executing Flag for port 1) is OFF in the following example, TXDU(256) outputs data through serial port 1 of the Serial Communications Unit with unit number 2. The 5 bytes of output data are read from the DM Area beginning at the rightmost byte of D100 and output through logical port 3 to a general-purpose device such as a printer.

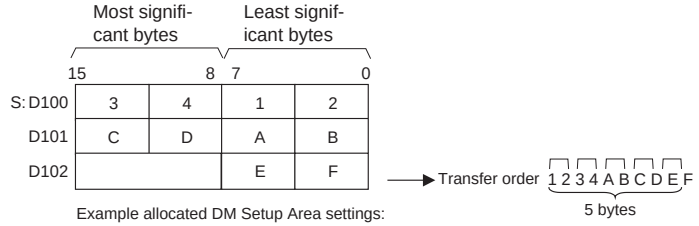






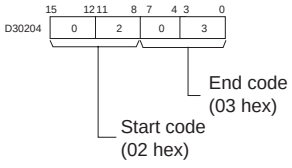
Note:
The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the Serial Communications Unit's unit address to the serial port's unit address.
(Set the unit address to 80 hex + 4 x unit number for port 1 or 81 hex + 4 x unit number for port 2.)



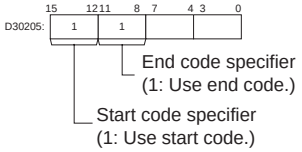


Example allocated DM Setup Area settings:

Start code and end code values



Start code and end code specifiers



In this example, a start and end code have been specified in the allocated DM Setup

ST	12	34	AB	CD	EF	ED

ST: Start code (e.g., 02 hex)
ED: End code (e.g., 03 hex)

Data sent.

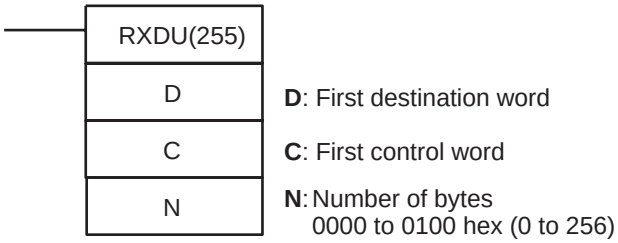
3-23-6 RECEIVE VIA SERIAL COMMUNICATIONS UNIT: RXDU(255)

Purpose

Reads the specified number of bytes of data from one of the CJ-series Serial Communications Unit's serial ports.

Note A CJ Unit Adapter is required to use CJ-series Serial Communications Units.

Ladder Symbol



Variations

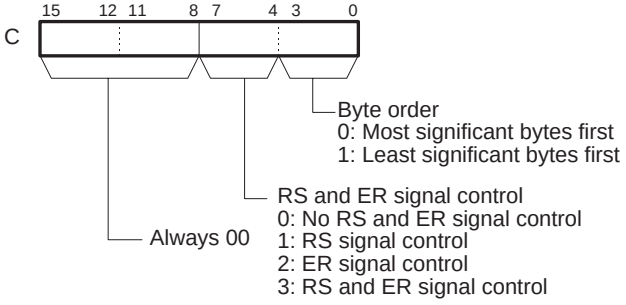
Variations	**Executed Each Cycle for ON Condition**	RXDU(255)
	Executed Once for Upward Differentiation	@RXDU(255)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

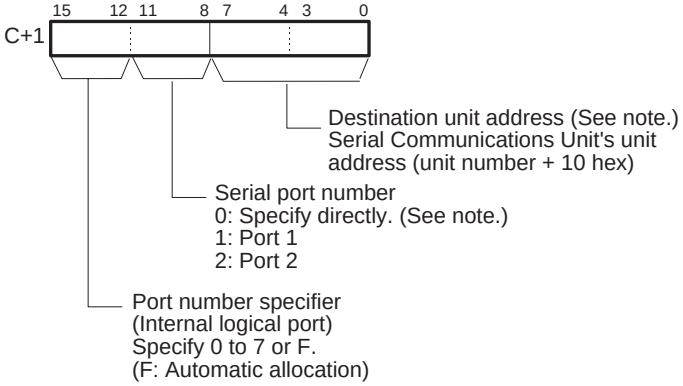
Applicable Program Areas

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

Operands

The contents of the control words, C and C+1, are as shown below.





Note The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the destination unit address to the serial port's unit address. (Set the destination unit address to 80 hex + 4 × unit number for port 1 or 81 hex + 4 × unit number for port 2.)

Operand Specifications

Area	D	C	D
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6142	CIO 0 to CIO 6143
Work Area	W0 to W511	W0 to W510	W0 to W511
Holding Bit Area	H0 to H511	H0 to H510	H0 to H511
Auxiliary Bit Area	A0 to A959	A0 to A958	A0 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D0 to D32767	D0 to D32766	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
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

RXDU(255) reads data that has been received in no-protocol mode at the Serial Communications Unit with the unit address specified in bits 0 to 7 of C+1, through the port specified with bits 8 to 11 of C+1, and stores that data starting at D. If fewer than N bytes of data have been received at the port, then only the data that has been received will be stored. The logical port number can be set to any value between 0 and 7 and is specified with bits 12 to 15 of C+1.

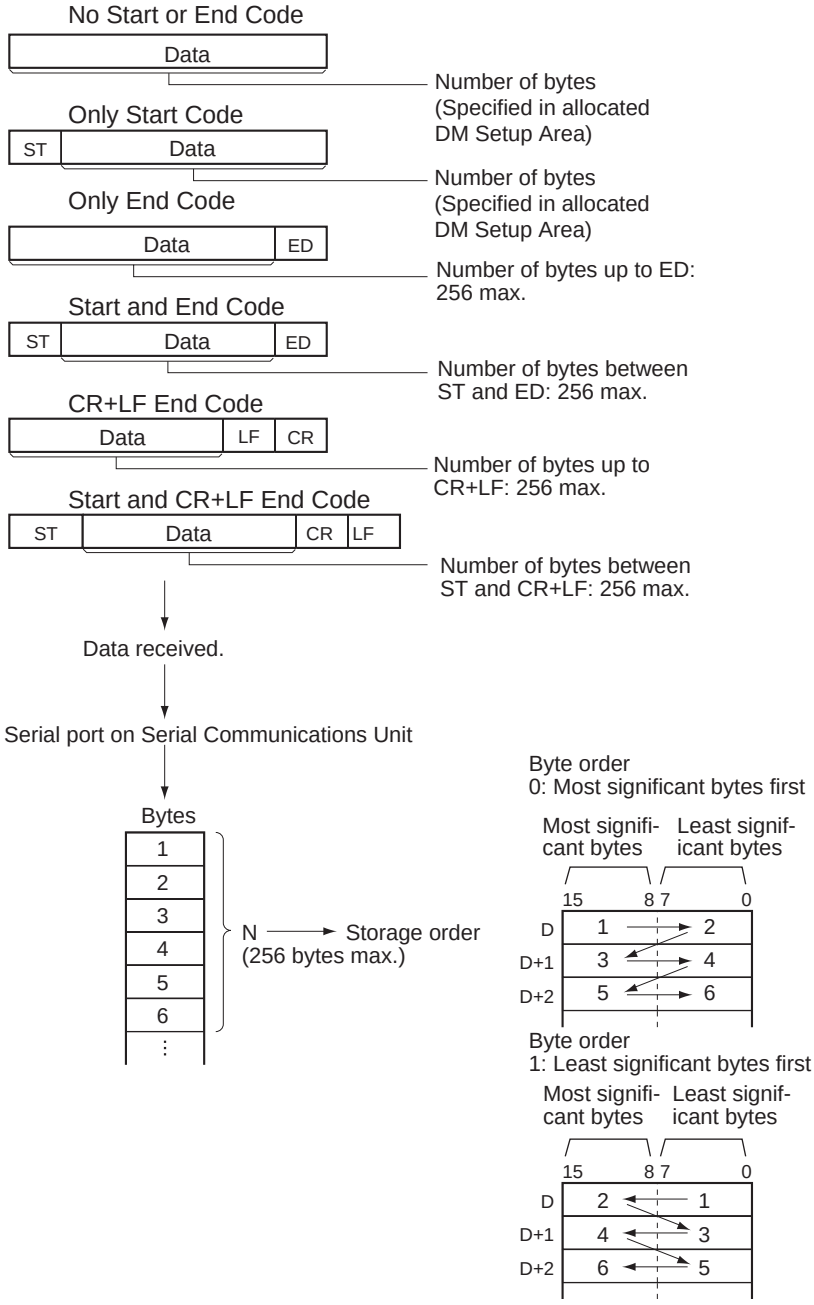
Execute RXDU(255) to read the received data from the buffer when the Reception Completed Flag (in the allocated DM Setup Area) is ON.

Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.

The following diagram shows the order in which data is received and the contents of the receive frame for various settings.

Note The logical port number can be allocated automatically by setting bits 12 to 15 of C+1 to F. For details, refer to *About Communications Port Numbers* on page 849.

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 all of the logical ports are being used or the Communications Port Enabled Flag for the specified logical port is OFF when the instruction is executed. 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

RXDU(255) can be used only for a Serial Communications Unit's serial port that has been set to no-protocol mode.

The following receive-message frame formats can be set in the allocated DM Setup Area.

- 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 be received is set from 00 to FF hex (1 to 256 decimal; 00 specifies 256 bytes).

The Reception Completed Flag will turn ON when the number of bytes specified the allocated DM Setup Area has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter will have the same value as the number of receive bytes specified in the allocated DM Setup Area. If more bytes are received than specified, the Reception Overflow Flag will turn ON.

If an end code is specified in the allocated DM Setup Area, the Reception Completed Flag 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 turns ON, the Reception Overflow Flag will turn ON.

Reception will be stopped if 259 bytes of data are received. If more data is input after that, the Overrun Error Flag and Transmission Error Flag will turn ON.

When more data is input to the Serial Communications Option Board's serial port than is specified in N, that data will be discarded when the next RXDU(255) instruction is executed.

When RXDU(255) is executed, data is stored in memory starting at D, the Reception Completed Flag will turn OFF (even if the Reception Overflow Flag (note 3) is ON), and the Reception Counter 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 and Reception Overflow Flag will be turned OFF, the Reception Counter 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.

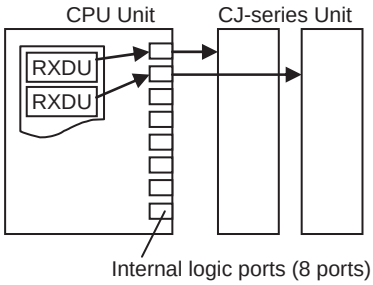
If CS and DR signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D and the status of the DR signal will be stored in bit 14 of D.

Receive data will not be stored if CS or DR signal monitoring is specified.

If 1, 2, or 3 hex is specified for RS and DR signal control in C, RXDU(255) will be executed regardless of the status of the Receive Completed Flag.

RXDU(255) uses a logical port (because it sends an internal FINS command) to output a receive sequence command to a Serial Communications Unit. Since SEND(090), RECV(098), CMND(490), PMCR(260), and TXDU(256) also use logical ports 0 to 7, RXDU(255) cannot be executed for a logical port if that logical port is already being used by one of those instructions or another RXDU(255) instruction.

To ensure that RXDU(255) is not executed while the logical port is busy, program the port's Communications Port Enabled Flag (A202.00 to A202.07) as a normally open condition.



RXDU(255) can not be executed while the Reception Completed Flag (bit 6 of n+9 or n+19, where n = CIO 1500 + 25 × unit number) is ON. Program the Reception Completed Flag as a normally open condition of RXDU(255).

An error will occur and the Error Flag will turn ON in the following cases.

- The Communications Port Enabled Flag for the specified logical port is OFF when RXDU(255) is executed.
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.

Read the data using RXDU(255) as soon as possible after the Reception Completed Flag turns ON. If reception is continued without reading the data, the reception buffer (capacity: 260 bytes) will overflow and data reception will stop. If this occurs, the port will have to be restarted to recover operation.

The reception buffer is cleared when RXDU(255) is executed for a serial port on the Serial Communications Unit. Thus, more than one RXDU(255) instruction cannot be used to read the data.

Related Flags and Words

The following words are related to RXDU(255) operation.

DM Setup Area Settings

(m = D30000 + 100 × unit number)

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
m+4	m+14	8 to 15	No-protocol Mode Start Code	00 to FF hex
		0 to 7	No-protocol Mode End Code	00 to FF hex
m+5	m+15	12 to 15	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0: None 1: Use end code. 2: Use CR+LF.

Auxiliary Area

Name	Address	Description
Communications Port Enabled Flags	A202.00 to A202.07	ON when a communications instruction (including RXDU(255)) can be executed with the corresponding port number. Bits 00 to 07 correspond to communications ports 0 to 7. The flag is OFF when a communications instruction is being executed and ON when the execution is completed (normal end or error end).
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding port numbers when communications instructions have been executed. Words A203 to A210 correspond to communications ports 0 to 7. The code is 00 while the instruction is being executed and contains the relevant code when execution is completed. These words are cleared to 0000 when PLC operation starts.
Communications Port Error Flags	A219.00 to A219.07	ON when an error occurred during execution of a communications instruction. When a flag is ON, check the completion code in A203 to A210 to troubleshoot the error. OFF when execution has been finished normally. Bits 00 to 07 correspond to communications ports 0 to 7. The flag status is retained until the next communications instruction is executed. Even if an error has occurred, a flag will be reset to 0 the next time that a communications instruction is executed for that port.

Completion Codes

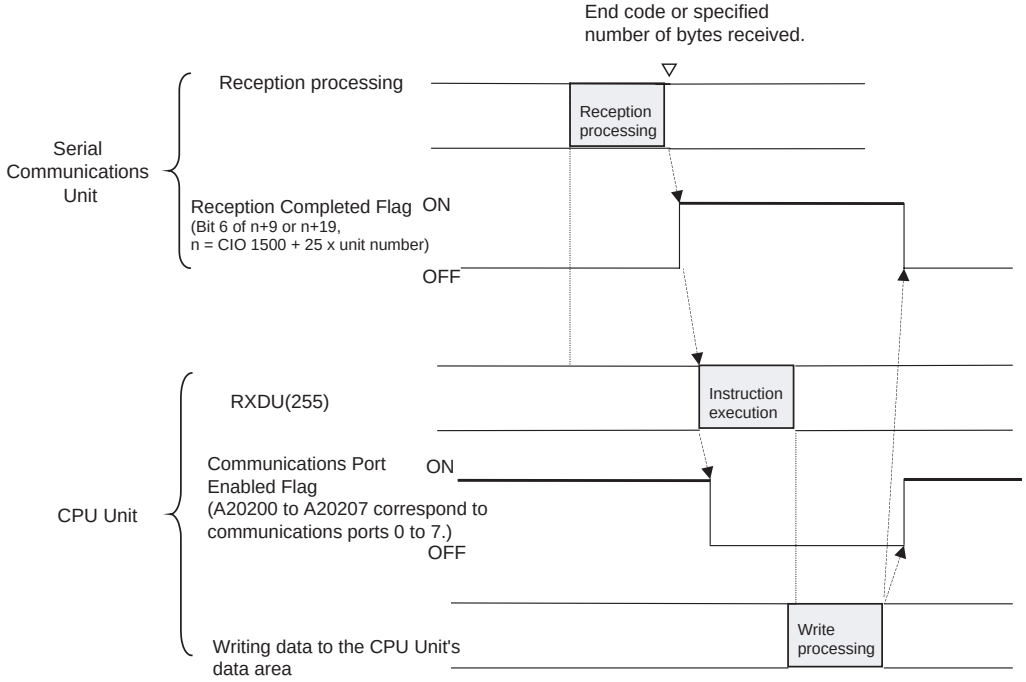
Code	Meaning
0205 hex	Response timeout (This error can occur when the communications mode is set to host link mode.)
0401 hex	Undefined command (This error can occur when the communications mode is set to protocol macro, NT Link, echoback test, or serial gateway mode.)
1001 hex	The command is too long.
1002 hex	The command is too short.
1004 hex	The command format is incorrect.
110C hex	Other parameter error
2201 hex	Operation could not be performed during operation. (Operation disabled because Unit is busy sending.)
2202 hex	Operation could not be performed when stopped. (Operation disabled because Unit is switching protocols.)

Related Flags in the CPU Bus Unit Area

(n = CIO 1500 + 25 × unit number)

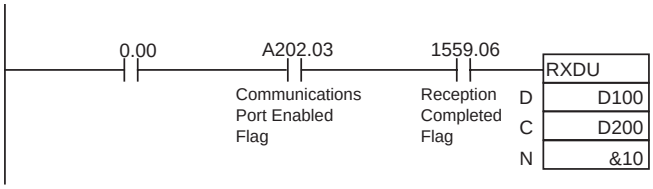
Word		Bit	Function
Port 1	Port 2		
n+8	n+18	04	Overrun Error Flag 1: The reception buffer contained more than 259 bytes of data before RXDU(255) was executed. Note: Once this error flag goes ON, it can be turned OFF only by turning the power OFF and then ON again or restarting the Board.
n+9	n+19	06	Reception Completed Flag 0: No data received or currently receiving data 1: Reception completed 0 → 1: The Board or Unit has received the specified number of bytes. 1 → 0: RXDU(255) was executed to write the data from the buffer to a CPU Unit data area.
n+9	n+19	07	Reception Overflow Flag 0: The Unit has not received more than the specified number of bytes. 1: The Unit has received more than the specified number of bytes. 0 → 1: The Unit received more data after data reception was completed. 1 → 0: RXDU(255) was executed to write the data from the buffer to a CPU Unit data area.
n+10	n+20	05	Reception Counter Indicates the number of bytes received in hexadecimal, between 0000 and 0100 hex (0 to 256 decimal).

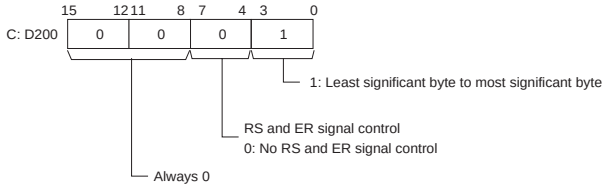
Example: Flag Operation The following diagram shows the operation of RXDU(255) and related flags.

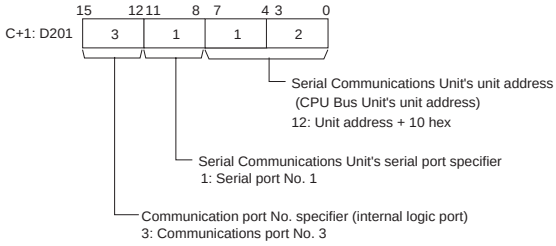


Example: Receiving Data

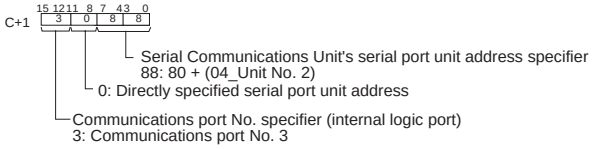
When CIO 0.00 is ON, A202.03 (the Communications Port Enabled Flag) is ON, and CIO 1559.06 (the Reception Completed Flag for port 1) is OFF in the following example, RXDU(255) reads the data received through serial port 1 of the Serial Communications Unit with unit number 2. (Logical communications port number 3 is used to receive the data from a general-purpose device such as a bar-code reader.) The 10 bytes of received data are written to the DM Area beginning at the rightmost byte of D100.

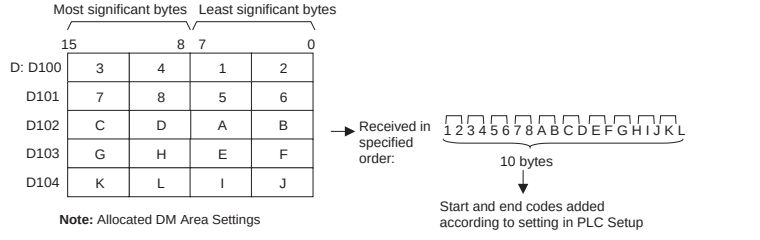






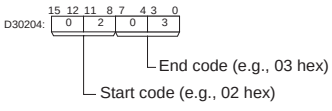
Note: The Serial Communications Unit's serial port unit address can also be directly specified in C+1.



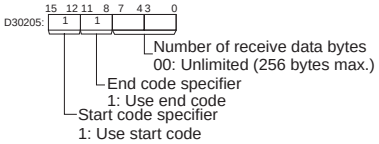


Note: Allocated DM Area Settings

- Start code/end code



- Start code/end code specifier

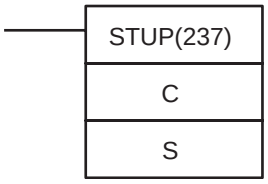


3-23-7 CHANGE SERIAL PORT SETUP: STUP(237)

Purpose

Changes the communications parameters of a serial port on a Serial Communications Option Board or CJ-series Serial Communications Unit (CPU Bus Unit). STUP(237) thus enables the protocol mode to be changed during PLC operation.

Ladder Symbol



C: Control word (port)

S: First source word

Variations

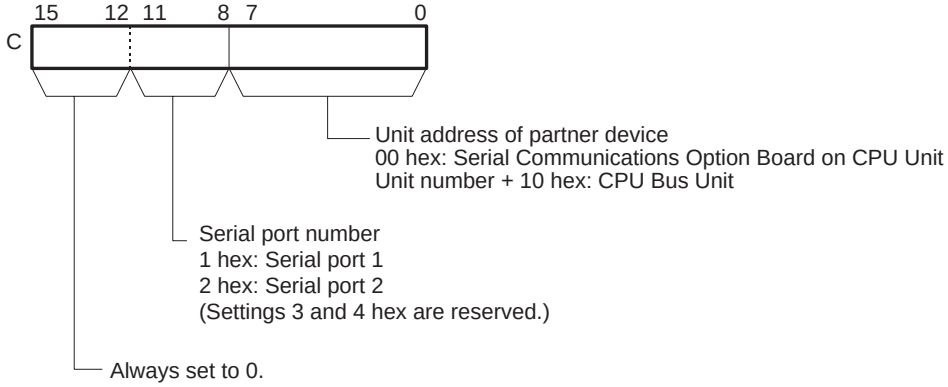
Variations		
	Executed Each Cycle for ON Condition	STUP(237)
	Executed Once for Upward Differentiation	@STUP(237)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

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 0 to CIO 6143	CIO 0 to CIO 6134
Work Area	W0 to W511	W0 to W502
Holding Bit Area	H0 to H511	H0 to H502
Auxiliary Bit Area	A0 to A438 A448 to A959	A0 to A438 A448 to A950
Timer Area	T0000 to T4095	T0000 to T4086
Counter Area	C0000 to C4095	C0000 to C4086
DM Area	D0 to D32767	D0 to D32758
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	Specified values only	#0000
Data Registers	DR0 to DR15	---

Area	C	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 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 Unit with the specified unit address, as shown in the following table. When the constant #0000 is designated to S, the communications settings of the corresponding port will be set to default.

Unit address	Unit	Port No.	Serial port	Serial port communications setup area
00 hex	CPU Unit	1 hex	Port 1	Communications parameters for the serial port 1 in the PLC Setup
		2 hex	Port 2	Communications parameters for serial port 2 in the PLC Setup
Unit No. + 10 hex	CJ-series Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1	10 words starting from D30000 + 100 x Unit No.
		2 hex	Port 2	10 words starting from D30000 + 100 x Unit No. + 10

Note Serial port 1 is the port on the Serial Communications Option Board mounted in option slot 1 and serial port 2 is the port on the Serial Communications Option Board mounted in option slot 2.

When STUP(237) is executed, the corresponding Port Parameters Changing Flag (A619.01, A619.02, or A619 to A636) 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 to change to Host Link communications for monitoring and programming from a host computer when specified conditions are meet while execution a communications sequence for a modem connection.

If the PLC is turned OFF and then ON again after STUP(237) has been used to change the communications parameters, the parameters will revert to the previous parameters, i.e., the ones before STUP(237) was executed.

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 Parameter Changing Flag is already ON. ON if STUP(237) is executed in an interrupt task. OFF in all other cases.

Precautions

Communications parameters consist of the protocol mode, baud rate, data format (protocol macro transmission method and protocol macro maximum communications data length), and other parameters. Refer to *CJ-series Serial Communications Boards and Serial Communications Unit Operation Manual (W336)* for the serial port that is to be set for details.

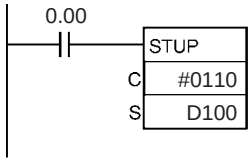
Related Flags and Words

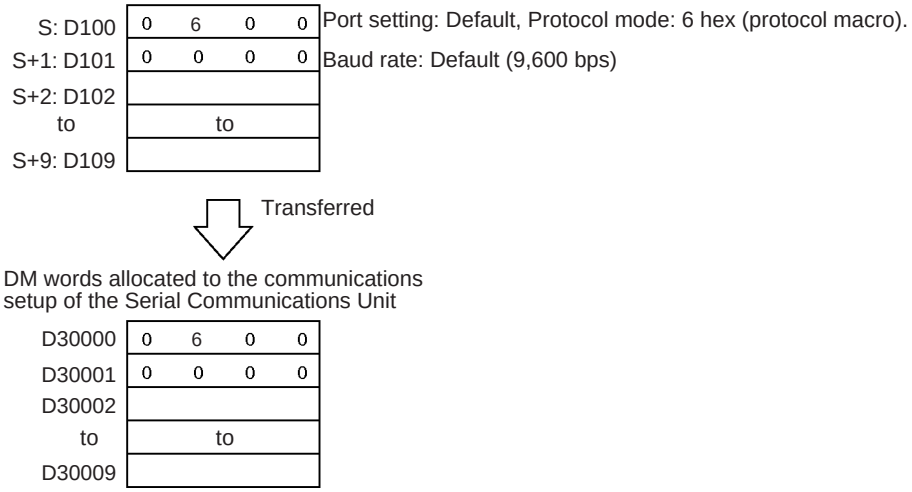
The following flags can be used as required when executing STUP(237). These flags are in the Auxiliary Area.

Name	Address	Contents
Serial Port 1 Parameters Changing Flag	A619.01	ON when the communications parameters are being changed for serial port 1.
Serial Port 2 Parameters Changing Flag	A619.02	ON when the communications parameters are being changed for serial port 2.
Port Parameters Changing Flags for ports 1 to 4 on Serial Communications Units 1 to 15.	A620.01 to A620.04 A635.01 to A635.04	ON when the communications parameters are being changed for a port on a Serial Communications Unit.

Examples

When CIO 0.00 turns ON in the following example, the communications parameters for serial port 1 of the Serial Communications Unit will be changed to the settings contained in the 10 words from D100 to D109. In this example, the setting are changed the protocol mode to the protocol macro mode.





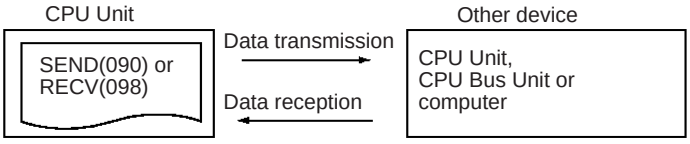
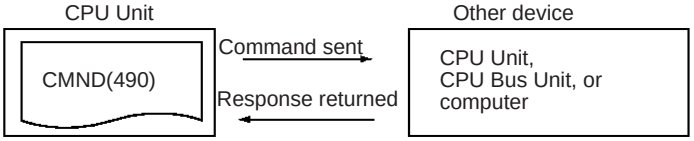
3-24 Network Instructions

This section describes instructions used to send and receive data via networks.

Instruction	Mnemonic	Function code	Page
NETWORK SEND	SEND	090	863
NETWORK RECEIVE	RECV	098	869
DELIVER COMMAND	CMND	490	875
EXPLICIT MESSAGE SEND	EXPLT	720	882
EXPLICIT GET ATTRIBUTE	EGATR	721	889
EXPLICIT SET ATTRIBUTE	ESATR	722	896
EXPLICIT WORD READ	ECHRD	723	901
EXPLICIT WORD WRITE	ECHWR	724	905

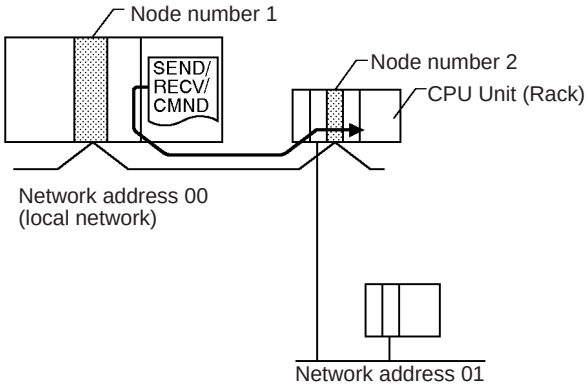
3-24-1 About Network Instructions

The network instructions are used between nodes on networks configured using Serial Communications Option Boards and CJ-series Units to transfer data and control operation, such as changing the operating mode. Nodes can consist of CP1H CPU Units, CS/CJ-series CPU Units, CS/CJ-series CPU Bus Units, and computers.

Instruction	Message content	Operation
SEND(090)/ RECV(098)	Commands to transmit/ receive data (FINS command)	
CMND(490)	Arbitrary commands (FINS command)	

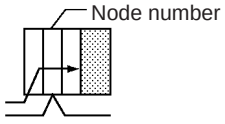
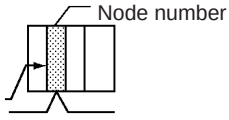
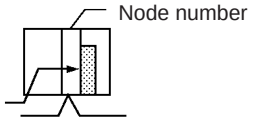
The commands executed by the network instructions are known as “FINS commands” and are used for communications between FA control devices. (Refer to the *CS/CJ/CP Series Communications Commands Reference Manual* for details on FINS commands.) With FINS commands it is possible to communicate (by the command/response format) with any Unit in any network on just by specifying the network address, node number, and unit number of the destination Unit.

In the following example, a FINS command is sent to the CPU Unit through node number 2 in network address 00.

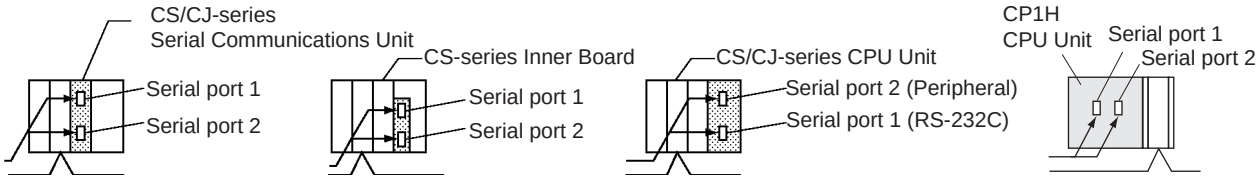


1,2,3...

1. Network address:
   Address of the network (local network = 00)
2. Node number
   Logical address in the network
3. Unit number
   Unit number of the destination Unit
   a. CP1H or CS/CJ-series CPU Unit: 00
   b. CS/CJ-series CPU Bus Unit: Unit number + 10 hexadecimal
   c. CS/CJ-series Special I/O Unit: Unit number + 20 hexadecimal
   d. CS-series Inner Board: E1 hexadecimal
   e. Computer:01

Unit number (hexadecimal)	Destination device
00	
Unit number +10	
E1	
01	

Note It is also possible to directly specify a serial port (unit address) within the destination device.



Serial Port Addresses:

- CS/CJ-series Serial Communications Unit ports

  Port 1: 80 hex + 4 × unit number

Unit number	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hexadecimal	80	84	88	8C	90	94	98	9C	A0	A4	A8	AC	B0	B4	B8	BC
Decimal	128	132	136	140	144	148	152	156	160	164	168	172	176	180	184	188

  Port 2: 81 hex + 4 × unit number

Unit number	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hexadecimal	81	85	89	8D	91	95	99	9D	A1	A5	A9	AD	B1	B5	B9	BD
Decimal	129	133	137	141	145	149	153	157	161	165	169	173	177	181	185	189

- CS-series Serial Communications Board ports

  Port 1: E4 hex (228 decimal)
  Port 2: E5 hex (229 decimal)

- CS/CJ-series CPU Unit ports

  Peripheral port: FD hex (253 decimal)
  RS-232C port: FC hex (252 decimal)

- CP1H CPU Unit

  Port 1: FD hex (253 decimal)
  Port 2: FC hex (252 decimal)

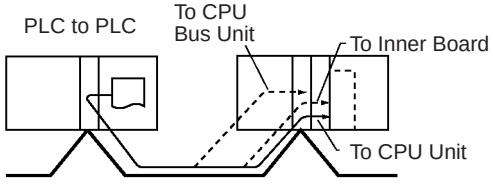
Note Serial port 1 is the port on the Serial Communications Option Board mounted in option slot 1 and serial port 2 is the port on the Serial Communications Option Board mounted in option slot 2.

Network Communications Patterns

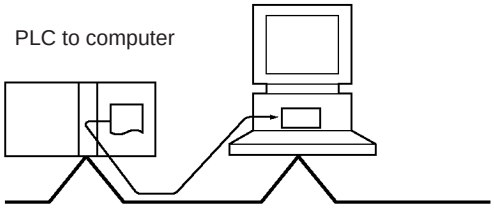
The following examples show three types of network communications: communications from a PLC to other devices in a network, communications from a PLC to serial ports on other devices in a network, and communications to a host computer connected by a Host Link.

Communications to Another Device in the Network

The following example shows communications from a PLC to devices in another PLC (the CPU Unit, CPU Bus Unit, or Inner Board). For more details, refer to the operation manual for the network (Controller Link or Ethernet) being used.

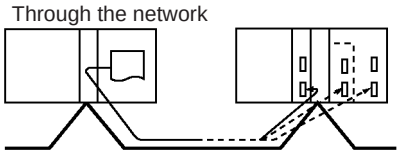


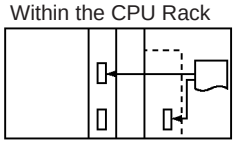
This example shows communications from a PLC to a personal computer.



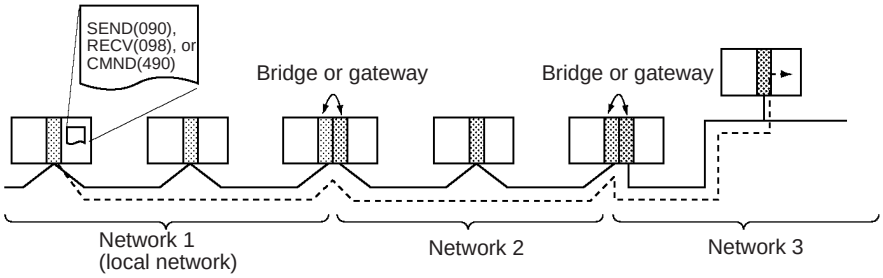
Communications to a Serial Port in the Network

These examples show communications from a PLC to serial ports in devices in the network. The first shows communications to serial ports in devices in another PLC (the CPU Unit, CPU Bus Unit, or Inner Board) and the second shows communications to a serial port within the CPU Rack itself.





Note Communications can span up to 8 network levels, including the local network. (The local network is the network where the communications originate.)



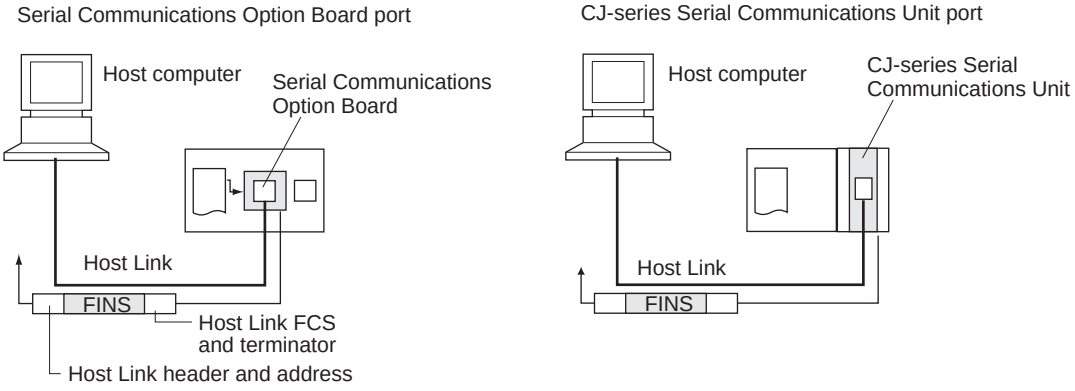
In order to communicate through the network, it is necessary to register a routing table in each PLC's CPU Unit which indicates the route by which data will be transferred to the desired node. Each routing table is made up of a local network table and a relay network table.

1,2,3...

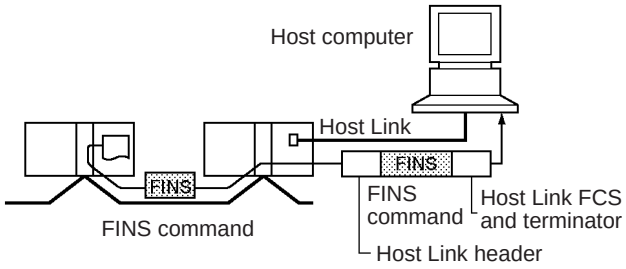
1. Local network table
   This table shows the unit numbers and network addresses of the nodes connected to the local PLC.
2. Relay network table
   This table shows the node numbers and network addresses of the first relay nodes to destination networks that are not connected to the local PLC.

Communications to a Host Computer (Host Link)

By sending a SEND(090), RECV(098), or CMND(490) instruction to a serial port set to Host Link mode, the necessary Host Link header and terminator will be attached to the FINS command and the command will be sent to the host computer.



Note Host Link communications can be sent through the network. In this case, the FINS command travels through the network normally. When the command reaches the Host Link system, the necessary Host Link header and terminator are attached to the FINS command and the command is sent to the host computer.

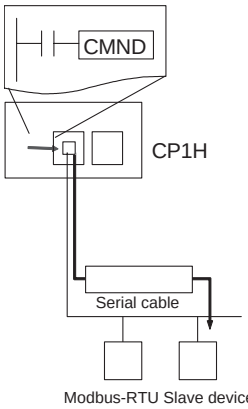


Serial Gateway Communications to a Component or Host Link Slave

It is possible to send FINS commands (or send/receive data) to a component or Host Link Slave connected to the PLC through its serial port with the serial gateway function.

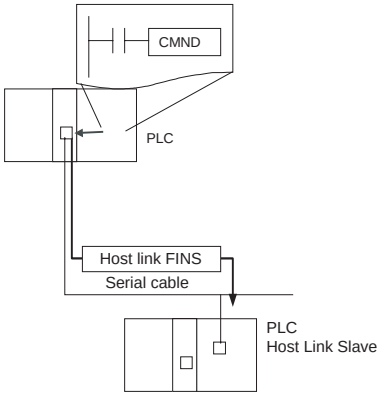
- Sending to a Component

  When a CMND(490) instruction is executed to a serial port that supports the serial gateway function, the serial gateway function converts the command to a CompoWay/F, Modbus-RTU, or Modbus-ASCII command.



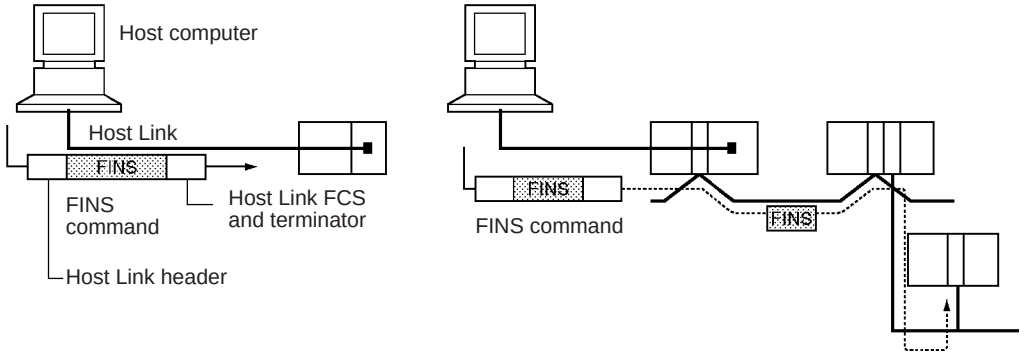
- Sending to a PLC operating as a Host Link Slave

  When a CMND(490), SEND(090), or RECV(098) instruction is executed to a serial port that supports the serial gateway function, the serial gateway function can send any FINS command or send/receive data.



Communications from a Host Computer (Host Link)

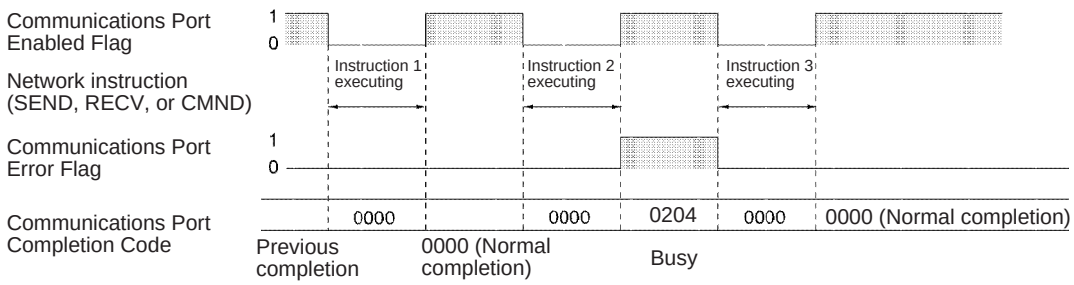
It is possible to send FINS commands from a host computer to the PLC to which it is connected as well as other devices in the network (CPU Units, Special I/O Units, computers, etc.). In this case, the necessary Host Link header and terminator must be attached to the FINS command when it is sent.



Communications Flags

The operation of the communications flags is outlined below.

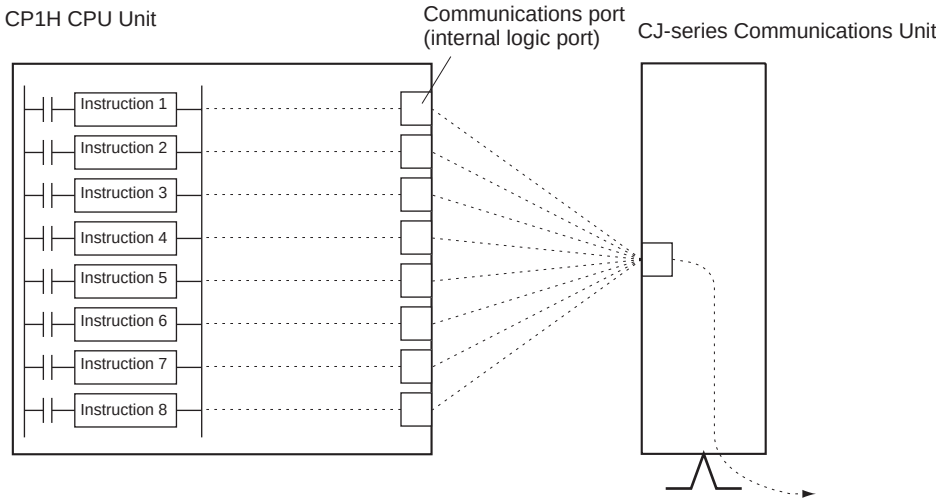
- The Communications Port Enabled Flag is reset to 0 when communications are in progress and set to 1 when communications are completed (normally or not).
- The status of the Communications Port Error Flag is maintained until the next time that data is transmitted or received.
- The Communications Port Error Flag will be reset to 0 the next time that data is transmitted or received, even if there was an error in the previous operation.



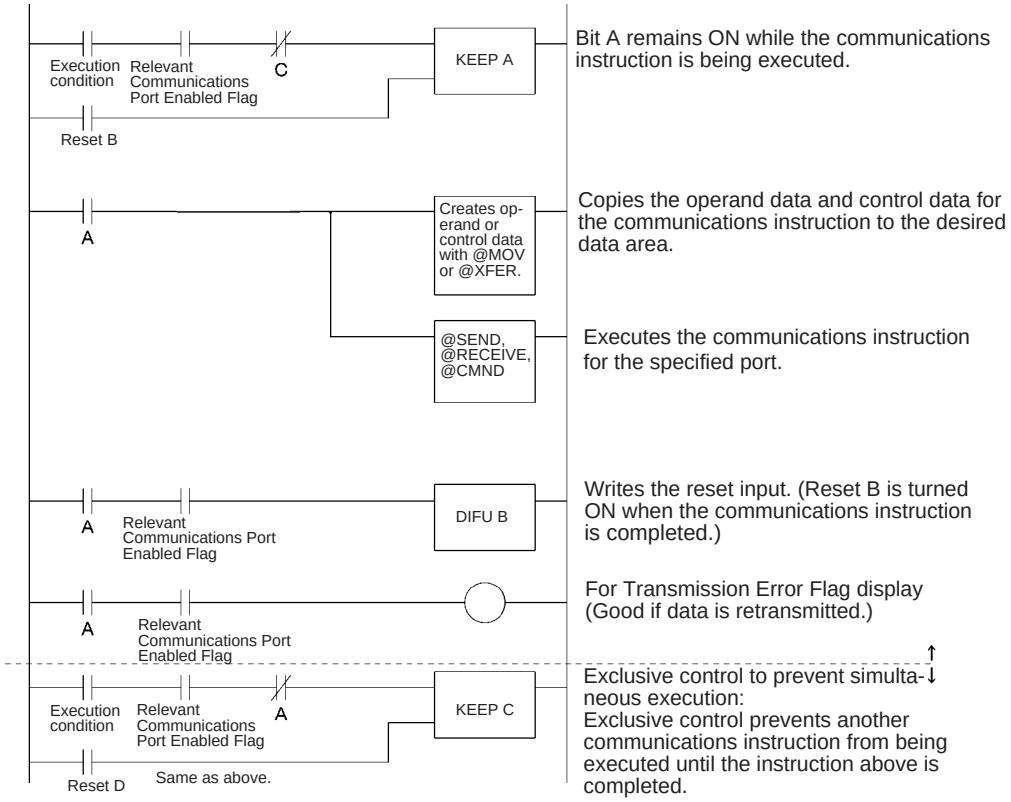
About Communications Port Numbers

There are 8 logical communications ports provided, so 8 communications instructions can be executed simultaneously. Only one instruction can be executed at a time for each communications port. Exclusive control must be used when more than 8 instructions are executed.

These 8 communications port numbers are shared by the network instructions (SEND(090), RECV(098), and CMND(490)), the serial communications instructions (TXDU(256) and RXDU(255)), and the PROTOCOL MACRO instruction (PMCR(260)). Be sure not to specify the same port number on two instructions at the same time.

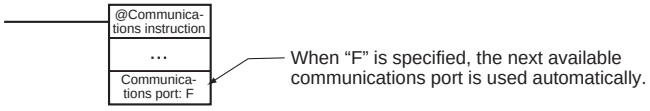


The following diagram shows an example of exclusive control.



Automatic Allocation of Communications Ports

The port number can be specified as “F” instead of from 0 to 7 to automatically allocate the communications port, i.e., the next open communications port is used automatically.



This saves the programmer from having to keep track of communications ports while programming. Even when automatic allocation is used, only a maximum of 8 ports can be used at the same time. Also, the communications port numbers must be used to access the flags and responses for specific communications ports by using A218 (Used Communications Port Number) and A216 and A217 (Network Communications Completion Code Storage Address).

The differences between assigning specific port numbers and automatically allocating port numbers are given in the following table.

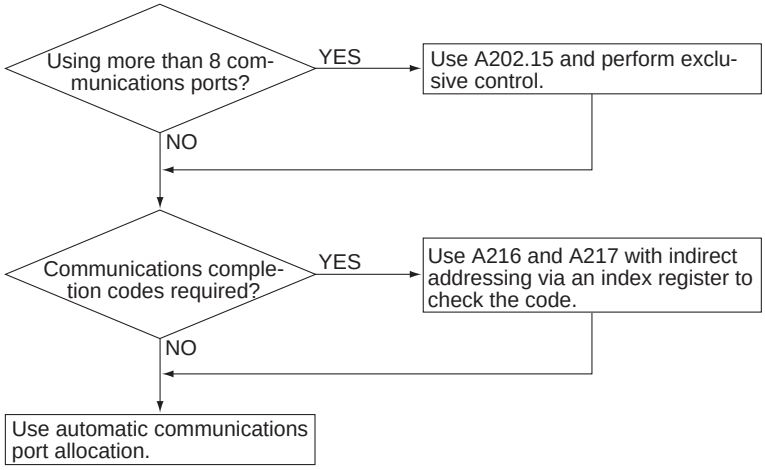
Item	Specific number assignments	Automatic allocation
Specification of the communications port number in the control data	0 to 7	F
Exclusive control	Required.	Not required unless more than 8 communications ports are required at the same time.
Flag applications	LD or LD NOT used with flag corresponding to the specified communications port.	TST(350) or TSTN(351) used with A218 (Used Communications Port Number).
Network communications completion codes	Completion code for communications port specified by user is accessed.	Completion codes are accessed by using the I/O memory address stored in A216 and A217 (Network Communications Completion Code Storage Address) and index register indirect addressing.

■ Auxiliary Area Bits and Words Used when Automatically Allocating Communications Ports

Address	Bits	Name	Description
A202	15	Network Communications Port Allocation Enabled Flag	ON when there is a communications port available for automatic allocation. This flag can be used to confirm if all eight communications ports have already been allocated before executing communications instructions.
A214	00 to 07	First Cycle Flags after Network Communications Finished	Each flag will turn ON for just one cycle after communications have been completed. Bits 00 to 07 correspond to ports 0 to 7. Use the Used Communications Port Number stored in A218 to determine which flag to access. **Note** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle.
	08 to 15	Do not use.	

Address	Bits	Name	Description
A215	00 to 07	First Cycle Flags after Network Communications Error	Each flag will turn ON for just one cycle after a communications error occurs. Bits 00 to 07 correspond to ports 0 to 7. Use the Used Communications Port Number stored in A218 to determine which flag to access. **Note** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle.
	08 to 15	Do not use.	
A216 and A217	---	Network Communications Completion Code Storage Address	The completion code for a communications instruction is automatically stored at the address with the I/O memory address given in these words. Place this address into an index register and use indirect addressing through the index register to reach the communications completion code.
A218	---	Used Communications Port Number	When a communications instruction is executed, the number of the communications port that was used is stored in this word. Values 0000 to 0007 hex correspond to communications ports 0 to 7.

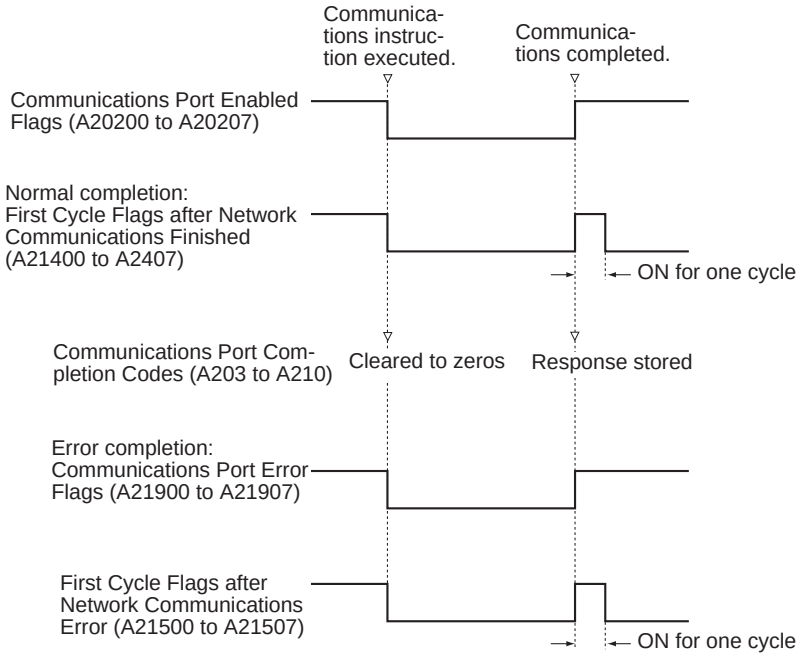
Note (1) Use the following flowchart to determine whether to use the Network Communications Port Allocation Enabled Flag (A20215) and the Network Communications Completion Code Storage Address (A216 and A217).



(2) The Auxiliary Area bits and words used for user-specified communications ports are listed in the following table.

Address	Bits	Name	Description
A202	00 to 07	Communications Port Enabled Flags	ON when a communications instruction can be executed with the corresponding port number. Bits 00 to 07 correspond to communications ports 0 to 7. The completion of communications can be confirmed by monitoring when a flag turns ON. The flag will turn OFF when execution of a communications instruction is started.
A203 to A210	---	Communications Port Completion Codes	These words contain the completion codes for the corresponding port numbers when communications instructions have been executed. Words A203 to A210 correspond to communications ports 0 to 7.
A219	00 to 07	Communications Port Error Flags	ON when an error occurred during execution of a communications instruction. When a flag is ON, check the completion code in A203 to A210 to troubleshoot the error. Turn OFF then execution has been finished normally. Bits 00 to 07 correspond to communications ports 0 to 7.

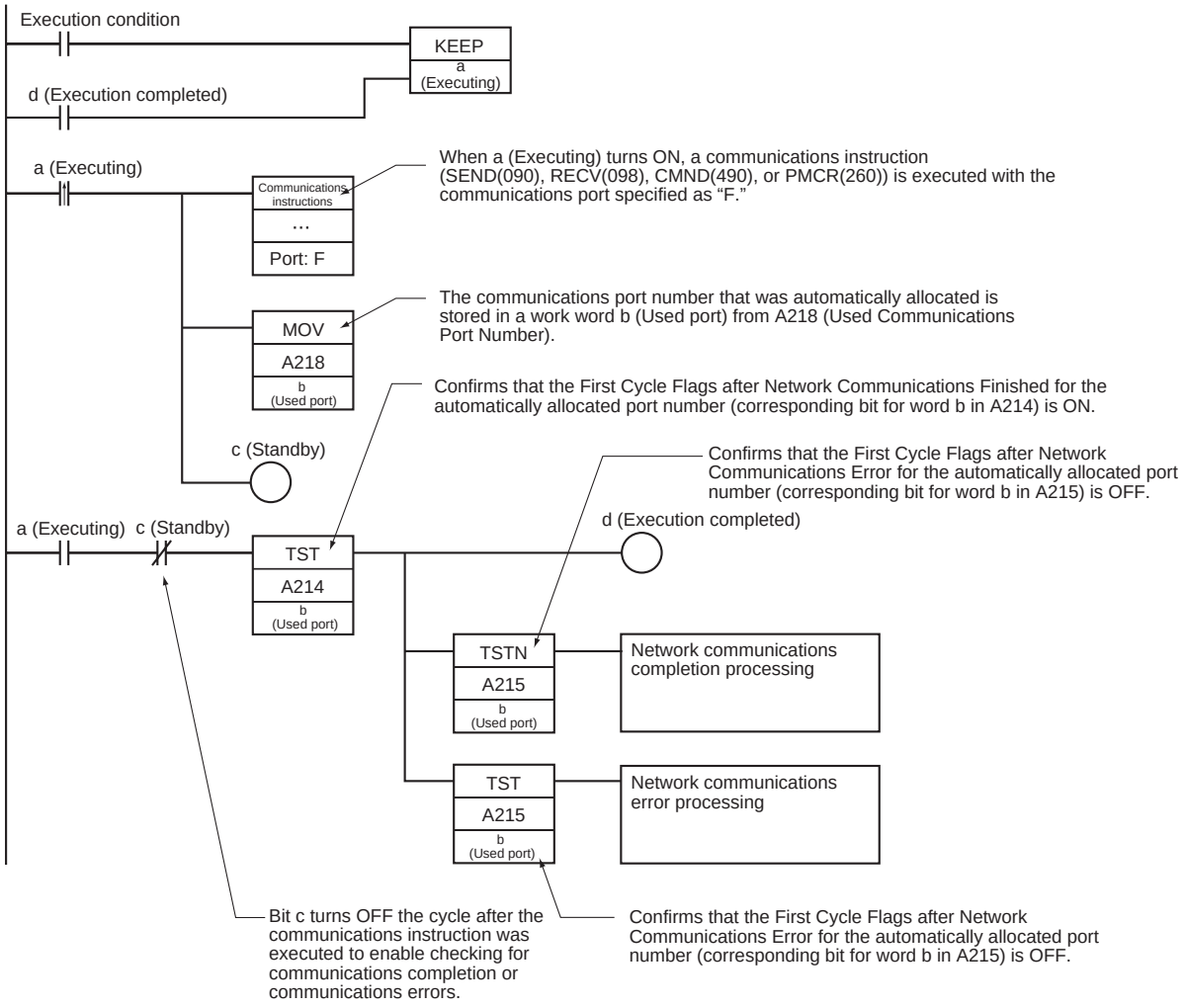
Flag/Word Operation



■ Applications Methods

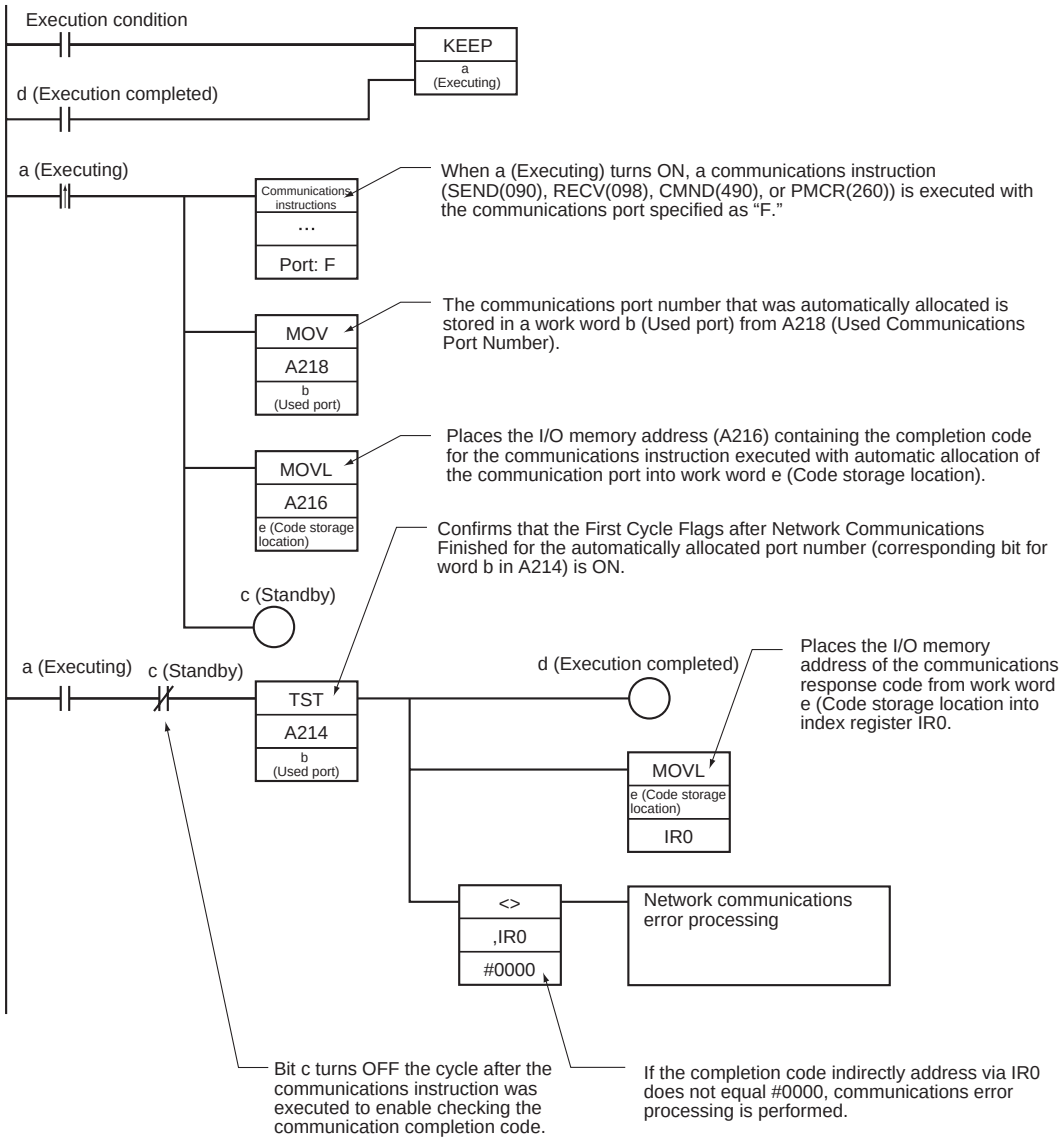
To use automatic communications port allocation, set the communications port number of “F” and then program as shown below.

Completing and Processing Error after Executing Communications Instructions



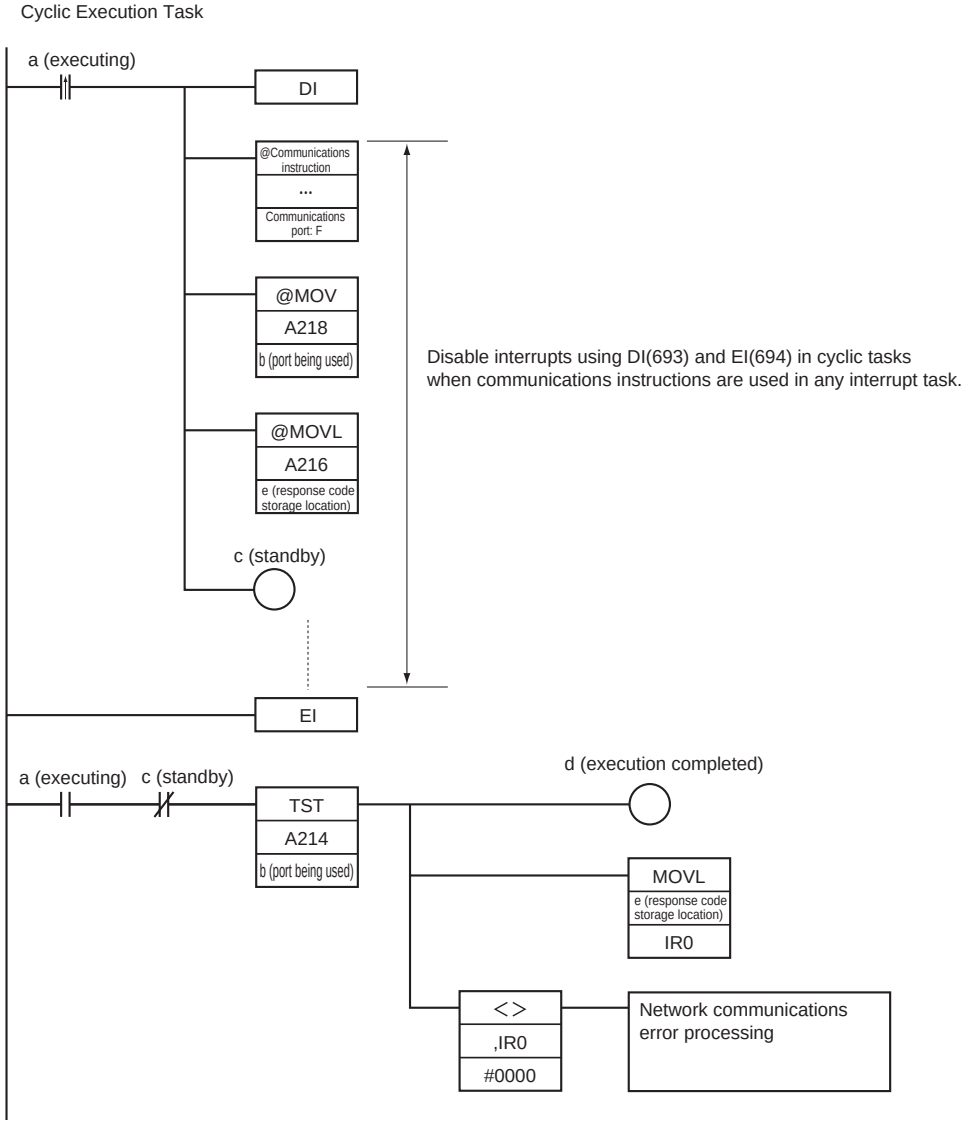
Accessing the Completion Code after Executing Communications Instructions

The completion codes are generally used to troubleshoot errors when they occur. A completion code of 0000 hex can, however, also be used to confirm that communications have completed normally.



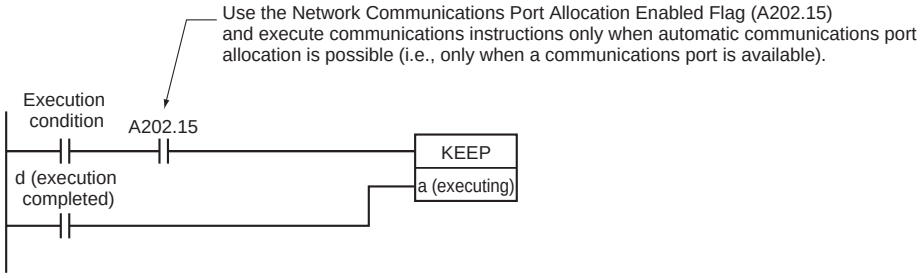
Using Communications Instructions Inside Interrupt Tasks

If communications instructions are used inside interrupt tasks, always use DI(693) and EI(694) to disable interrupts before and after communications instructions using automatically allocated communications port numbers in the cyclic tasks regardless of whether user-specified or automatically allocated communications port numbers are being used in the interrupt tasks. An example is shown below.



Preventing Exceeding the Maximum Number of Communications Ports

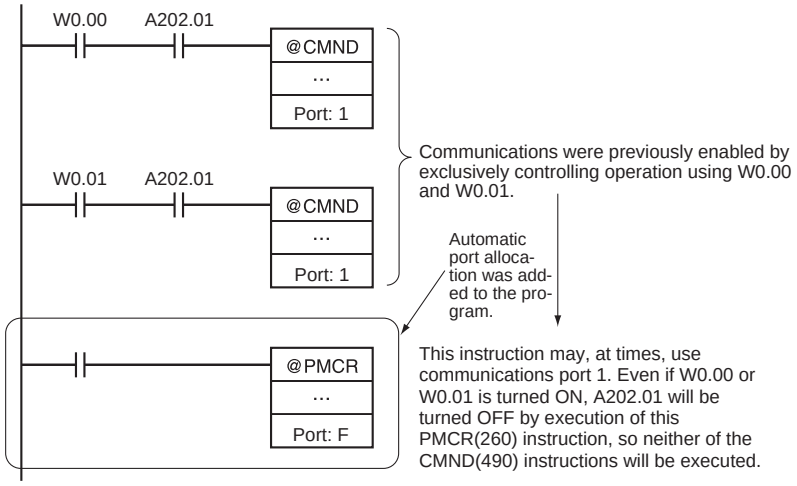
If there is a possibility of executing more than eight communications instructions at the same time, use the following type of programming to confirm if a communications port is available even when using automatically allocated communications port numbers.



Combining Automatic Port Specification with User-specified Ports

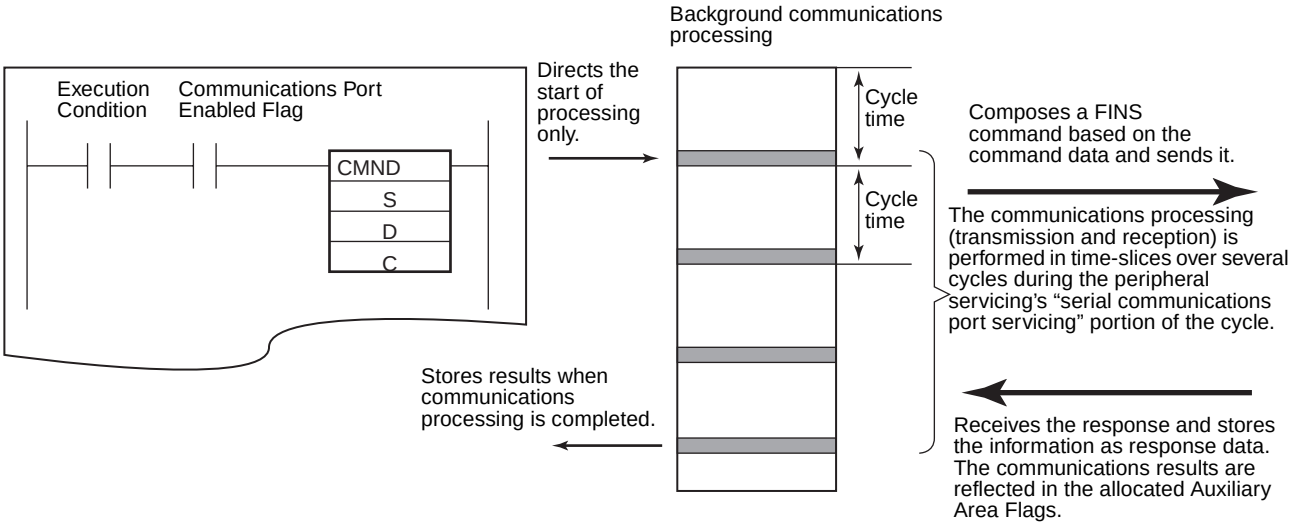
Both user-specified communications port numbers and automatically specified communications port numbers can be used in the same program. It is possible, however, that the communications port numbers specified by the user will be used for automatic allocation. It is thus important to check the program carefully when adding communications instructions that use automatic communications port allocation to an existing program, as shown in the following example.

Programming Example



Timing the Execution of Network Instructions

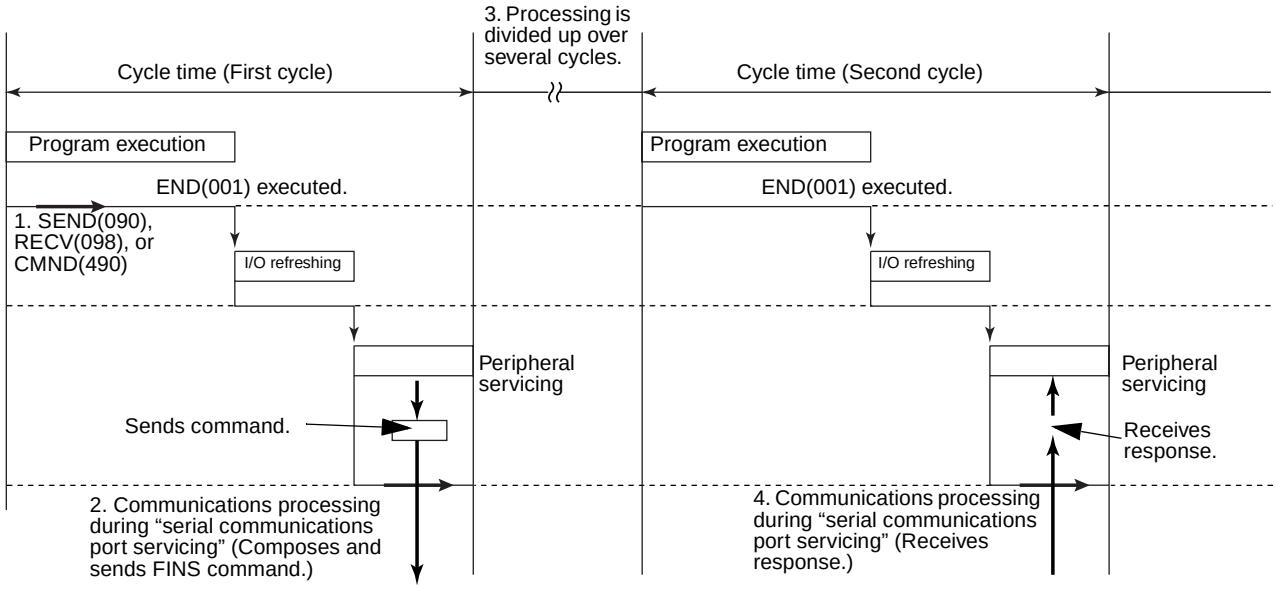
A Network Instruction just starts the communications processing when its execution condition is established. The actual communications processing is executed in the background in the "serial communications port servicing" portion of peripheral servicing.



The communications processing is performed as follows:

1. If the corresponding Communications Port Enabled Flag (A202.00 to A202.07) is ON when the execution condition is established, the system performs the following processes:
   - Turns OFF the port's Communications Port Enabled Flag and Communications Port Error Flag (A219.00 to A219.07).
   - Sets the port's Communications Port Completion Code (A203 to A210) to 0000.
   - Reads the control words (beginning at C) and starts communications processing (sending a FINS command or receiving a response.)
2. In the peripheral servicing's "serial communications port servicing" portion of the cycle, the system composes a FINS command based on the operands (see note) and sends the FINS command to the Communications Unit or other destination node.

   Note When SEND(090) is being executed, the contents of S and D are read and a FINS command for data transmission is composed.
   When RECV(098) is being executed, the content of S is read and a FINS command for data reception is composed.
   When CMND(490) is being executed, the content of S is read and the corresponding FINS command is composed.
3. If the send processing cannot be completed in a the time available in "serial communications port servicing" period, the processing will be continued in the next cycle's serial communications port servicing.
4. When a response is returned, the system performs the following processes:
   - Refreshes the destination words specified in the Network instruction with the response data.
   - Turns ON the port's Communications Port Enabled Flag.
   - Refreshes the port's Communications Port Error Flag (A219.00 to A219.07) and Communications Port Completion Code (A203 to A210).



3-24-2 About Explicit Message Instructions

Methods for Using Explicit Message Communications

There are two methods that can be used to send explicit messages from a PLC.

- Use the CMND(490) to send a FINS command code of 2801 hex (EXPLICIT MESSAGE SEND).
- Use the following Explicit Message Instructions.

Explicit Message Instructions

The following instructions are called Explicit Message Instructions.

Instruction	Name	Outline
EXPLT(720)	EXPLICIT MESSAGE SEND	Sends an explicit message with any service code. Note: Functionally, this instruction is the same as sending CMND(490) with a FINS command code of 2801 hex.
EGATR(721)	EXPLICIT GET ATTRIBUTE	Sends an explicit message with a service code of 0E hex (GET ATTRIBUTE SINGLE).
ESATR(721)	EXPLICIT SET ATTRIBUTE	Sends an explicit message with a service code of 10 hex (SET ATTRIBUTE SINGLE).
EGATR(721)	EXPLICIT WORD READ	Uses an explicit message to read data from a CPU Unit.
EGATR(721)	EXPLICIT WORD WRITE	Uses an explicit message to write data to a CPU Unit.

Features of Explicit Message Instructions

- Explicit Message Instructions do not require giving a 2801 hex FINS command and are much simpler to program than CMND(490).
- With the EXPLICIT GET/SET ATTRIBUTE instructions, entering the service code is not required and only information from the class ID onward needs to be entered.
- With the EXPLICIT WORD READ/WRITE instructions, the I/O memory address in the local and remote CPU Units can be specified directly.

  Code specifications for area types and hexadecimal word addresses are not required. (These are required for CMND(490) instructions with service code 1E (word data read) or 1F hex (word data write).)
  This enables easy reading and writing of data between CPU Units using explicit message communications (like SEND/RECV instructions for FINS commands).

Operation

The Explicit Communications Error Flag is used to determine if communications ended normally or in error.

For error completions (i.e., when the flag is ON), the Communications Port Error Flag for FINS commands is used to determine if the explicit message was never sent (i.e., when the flag is ON) or if there was an error in the explicit message that was sent (i.e., when the flag is OFF).

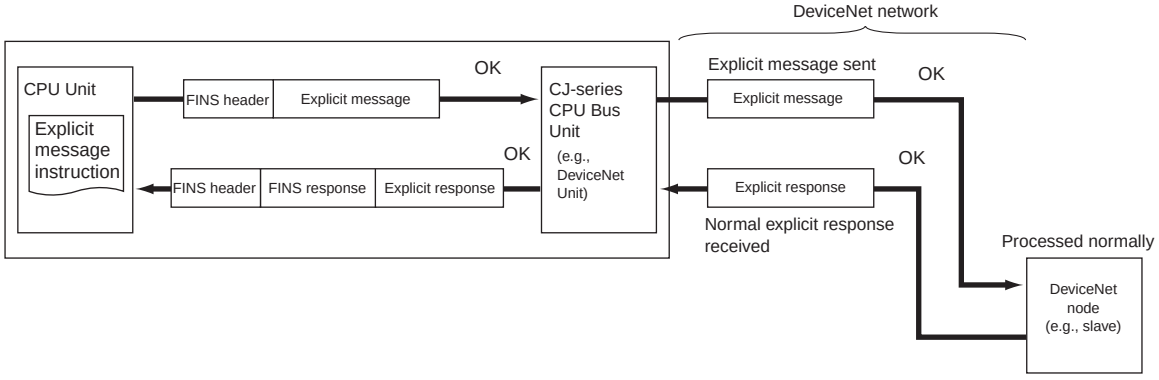
The Communications Port Completion Code will contain 0000 hex after a normal end, an explicit message error code after an explicit communications error end, and a FINS message completion code after a FINS error end.

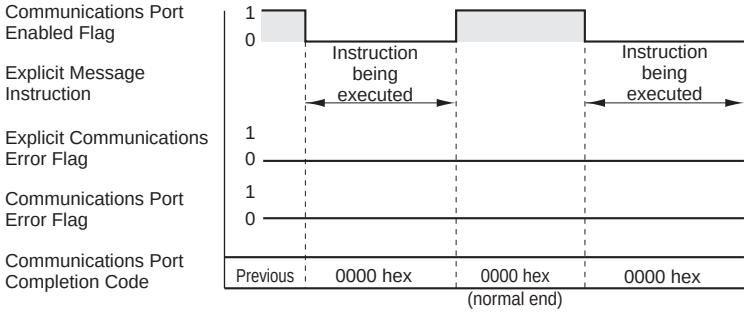
Condition		Explicit Communications Error Flag (A213.00 to A213.07: Communications port No. 0 to 7)	Communications Port Error Flag (A219.00 to A219.07: Communications port No. 0 to 7)	Communications Port Completion Code (A203 to A210: Communications port No. 0 to 7)
1) Normal end		OFF	OFF	0000 hex
2) Error end	a) When the explicit message could not be sent	ON	ON	FINS messages completion code
	b) When the explicit message was sent but an explicit error response was returned		OFF	Explicit message error code

1) Normal End

An explicit message is sent and a normal response is returned.

The corresponding Explicit Communications Error Flag (A213.00 to A213.07: Communications port No. 0 to 7) will be OFF and the Communications Port Completion Code (A203 to A210: Communications port No. 0 to 7) will contain the explicit message normal response code of 0000 hex.





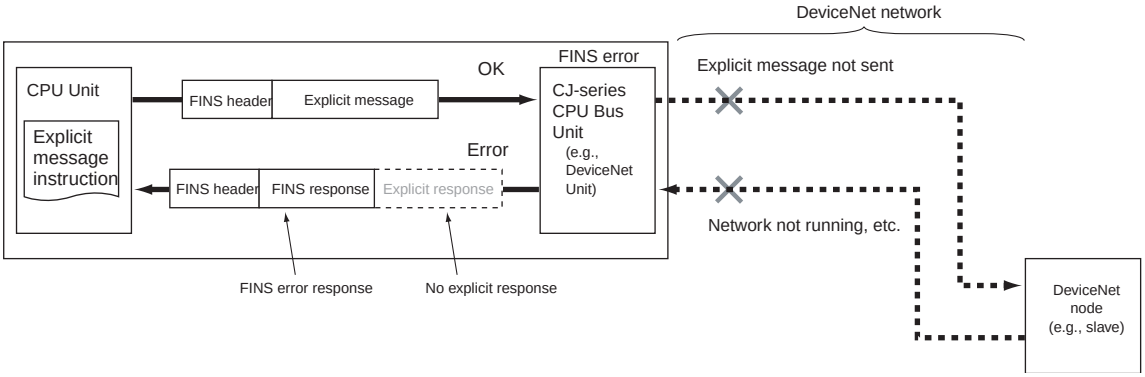
2) Error End

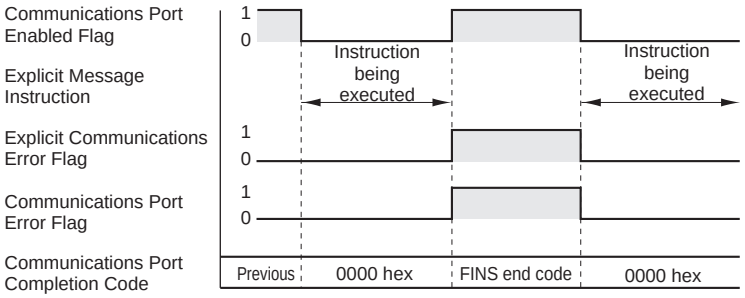
The are two possibilities for error ends, as described in the next two subsections.

a) When the Explicit Message Could Not Be Sent

In this case, the explicit message was never sent on the network, e.g., because the network was not running. Here, both the Explicit Communications Error Flag (A213.00 to A213.07: Communications port No. 0 to 7) and the Communications Port Error Flag (A219.00 to A219.07: Communications port No. 0 to 7) will turn ON.

After completion, the Communications Port Completion Code (A203 to A210: Communications port No. 0 to 7) will contain the FINS message error code.

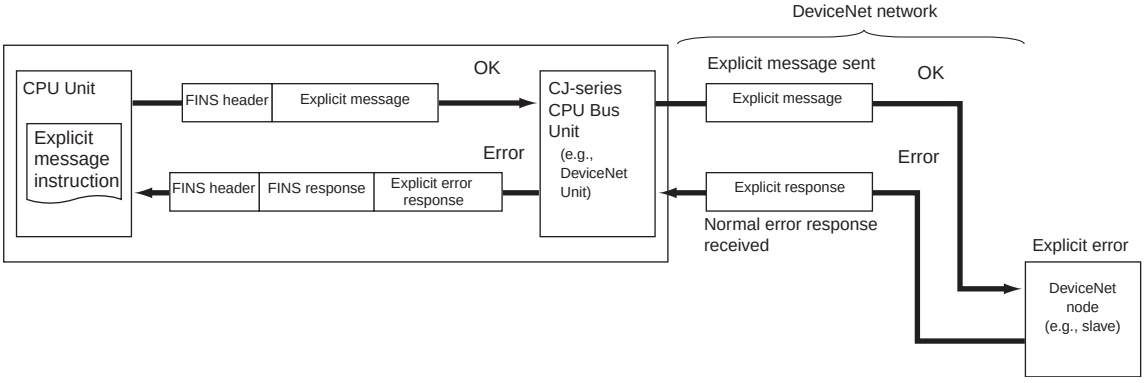


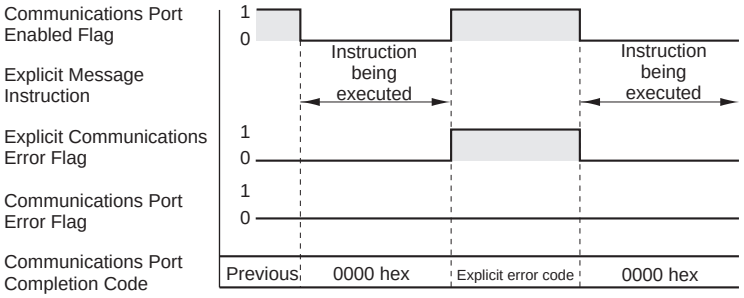


b) When the Explicit Message Was Sent But an Explicit Error Response Was Returned

In this case, the explicit message was sent but an error existed in the explicit message command frame (code not supported, illegal size, etc.). Here, the Explicit Communications Error Flag (A213.00 to A213.07: Communications port No. 0 to 7) will turn ON and the Network Communications Error Flag (A219.00 to A219.07: Communications port No. 0 to 7) will remain OFF.

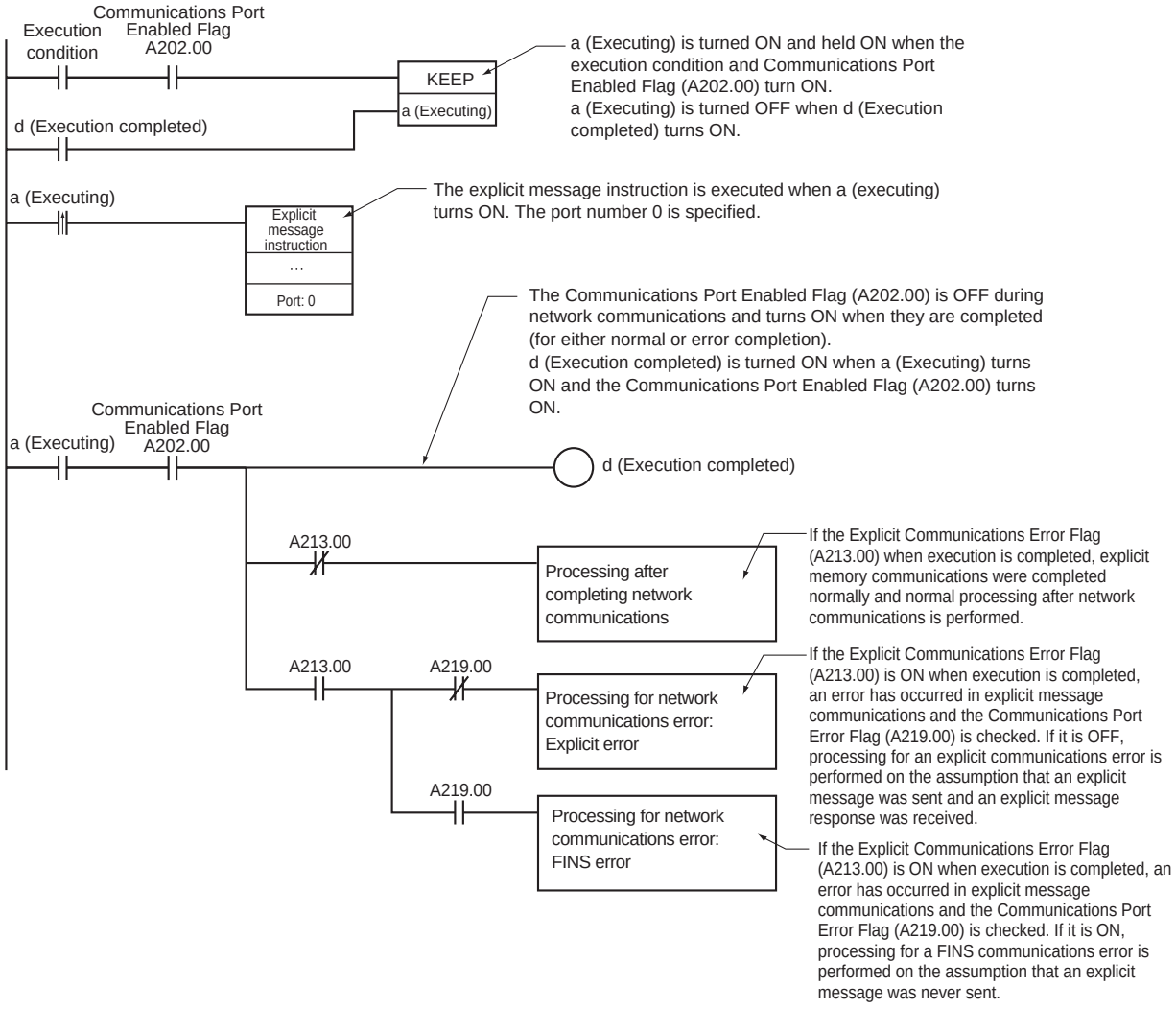
After completion, the Network Communications Response Code (A203 to A210: Communications port No. 0 to 7) will contain the explicit message error code.



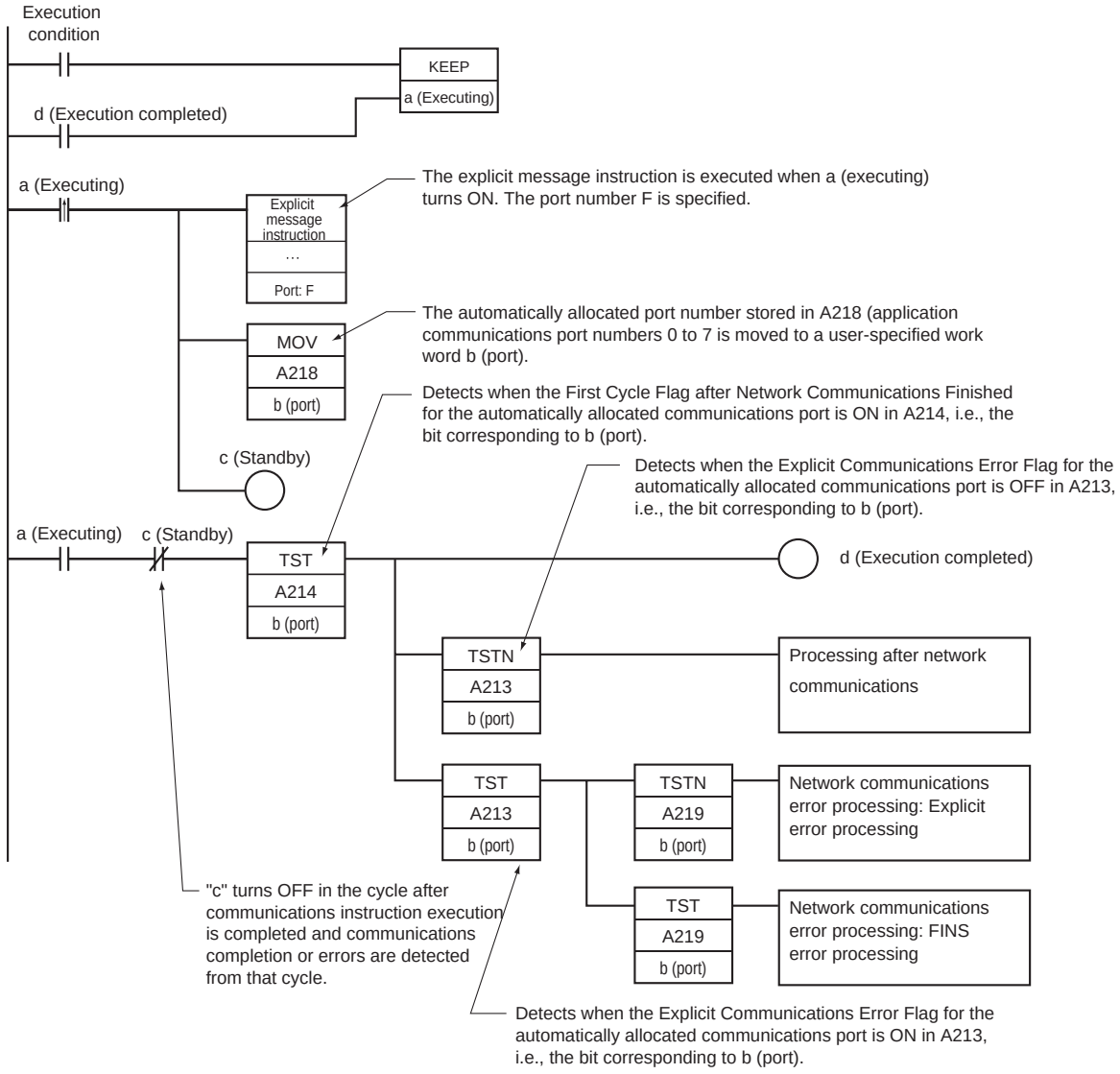


Ladder Programming Examples

Example 1: User Specification of Communications Port Number



Example 2: Automatic Allocation of Communications Port Number

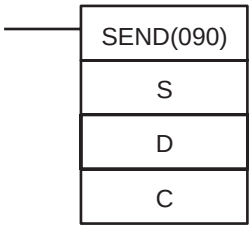


3-24-3 NETWORK SEND: SEND(090)

Purpose

Sends data to a node in the network.

Ladder Symbol



S: First source word (local node)

D: First destination word (remote node)

C: First control word

Variations

Variations	**Executed Each Cycle for ON Condition**	SEND(090)
	Executed Once for Upward Differentiation	@SEND(090)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: First control word

The five control words C to C+4 specify the number of words being transmitted, the destination, and other settings shown in the following table.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words: 0001 to maximum allowed[1] (4-digit hexadecimal)	
C+1	Destination network address: 00 to 7F (0 to 127)[2, 4]	Bits 08 to 11:Serial port number[3] (physical port) 1 hex: Port 1 2 hex: Port 2 (Do not set 0, 3, or 4.) Bits 12 to 15: Always 0.
C+2	Destination unit address: 00 to FE[5]	Destination node address: 00 to maximum allowed[6]
C+3	No. of retries: 00 to 0F (0 to 15)	Bits 08 to 11: Communications port number (internal logic port): 0 to 7, Automatic allocation: F[7] Bits 12 to 15: Response setting 0: Response requested. 8: No response requested.[8]
C+4	Response monitoring time: 0001 to FFFF (0.1 to 6553.5 seconds) (The default setting of 0000 sets a monitoring time of 2 seconds.)	

Note

(1) The maximum number of words allowed depends on the network being used. For a Controller Link, the allowed range is 0001 to 03DE (1 to 990 words).

(2) Set the destination network address to 00 to transmit within the local network. When two or more CPU Bus Units are mounted, the network address will be the unit number of the Unit with the lowest unit number.

(3) The following two methods can be used to send data to the host computer through a serial port with the host link while initiating communications from the PLC.

(a) Set the destination unit address (bits 00 to 07 of C+2) to the unit address of the CPU Unit or CJ-series Serial Communications Unit and set the serial port number (bits 08 to 11 of C+1) to 1 for port 1 or 2 for port 2.

Unit address (C+2, bits 00 to 07)	Unit	Serial port number (C+1, bits 08 to 11)	Serial port
00 hex	CP1H with Serial Communications Option Board mounted	1 hex	Port 1
		2 hex	Port 2
10 hex + unit number	CJ-series Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1
		2 hex	Port 2

(b) Set the destination unit address directly into bits 00 to 07 of C+2. In this case, set the serial port number in bits 08 to 11 of C+1 to 0 for direct specification.

CPU Unit Ports on Serial Communications Option Boards

Port	Port's unit address
Port 1	FD hex (253 decimal)
Port 2	FC hex (252 decimal)

CJ-series Serial Communication Unit Ports

Port	Port's unit address	Example: Unit number = 1
Port 1	80 hex + 4 × unit number	80 + 4 × 1 = 84 hex (132 decimal)
Port 2	81 hex + 4 × unit number	81 + 4 × 1 = 85 hex (133 decimal)

(4) When specifying the serial port without a routing table for the serial gateway function (conversion to host link FINS), set the serial port's unit address in the destination network address byte.

(5) The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
CP1H CPU Unit (Serial Communications Option Board) or CJ-series CPU Unit	00 hex
CS/CJ-series CPU Bus Unit	10 hex + unit number
CS/CJ-series Special I/O Unit	20 hex + unit number
CS-series Inner Board	E1 hex
Computer	01 hex
Unit connected to network (not necessary to specify Unit)	FE hex
Direct specification of the serial port's unit address	CP1H CPU Unit with Serial Communications Option Board Mounted Port 1: FD hex (253 decimal) Port 2: FC hex (252 decimal) CS/CJ-series Serial Communications Unit ports Port 1: 80 hex + 4 × unit number Port 2: 81 hex + 4 × unit number CS-series Serial Communications Board ports Port 1: E4 hex (228 decimal) Port 2: E5 hex (229 decimal) CS/CJ-series CPU Unit ports Peripheral port: FD hex (253 decimal) RS-232C port: FC hex (252 decimal)

(6) The maximum node number depends on the network being used. For a Controller Link, the allowed range is 00 to 20 hexadecimal (0 to 32). Set the destination node number to FF to broadcast to all nodes; set it to 00 to transmit within the local node.

(7) Refer to *Automatic Allocation of Communications Ports* on page 850 for details on using automatic allocation of the communications port number (logical port).

(8) When the destination node number is set to FF (broadcast transmission), there will be no response even if bits 12 to 15 are set to 0.

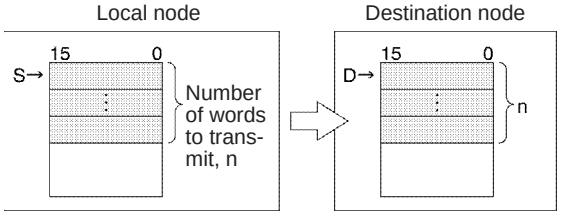
Operand Specifications

Area	S	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6139
Work Area	W0 to W511		W0 to W507

Area	S	D	C
Holding Bit Area	H0 to H511		H0 to H507
Auxiliary Bit Area	A0 toA959		A0 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D0 to D32767		D0 to D32763
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SEND(090) transfers the data beginning at word S to addresses beginning at D in the designated device through the PLC's CPU Bus or over a network. The number of words to be transmitted is specified in C.



If the destination node number is set to FF, the data will be broadcast to all of the nodes in the designated network. This is known as a broadcast transmission.

If a response is requested (bits 12 to 15 of C+3 set to 0) but a response has not been received within the response monitoring time, the data will be retransmitted up to 15 times (retries set in bits 0 to 3 of C+3). There will be no response or retries for broadcast transmissions.

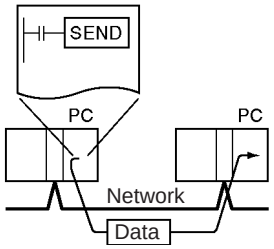
SEND(090) can be used to transmit data to a particular serial port in the destination device as well as the device itself.

Data can be transmitted to a host computer connected to the PLC's serial port (when set to host link mode) as well as a PLC or computer connected through a Controller Link or Ethernet network.

If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when SEND(090) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A202.00 to A202.07) and Communications Port Error Flag (ports 00 to 07: A219.00 to A219.07) will be turned OFF and 0000 will be written to the word that contains the completion code (ports 00 to 07: A203 to A210). Data will be transmitted to the destination node once the flags have be set.

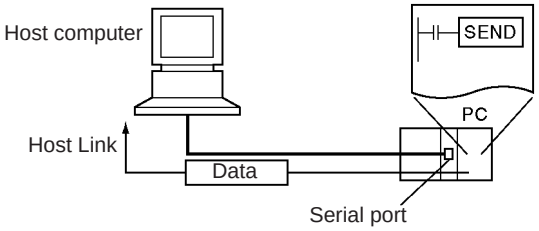
Transmission through the Network

SEND(090) can be used to transmit data from the PLC to the specified data area in a PLC or computer connected by a Controller Link network or Ethernet link.



Transmission through Host Link

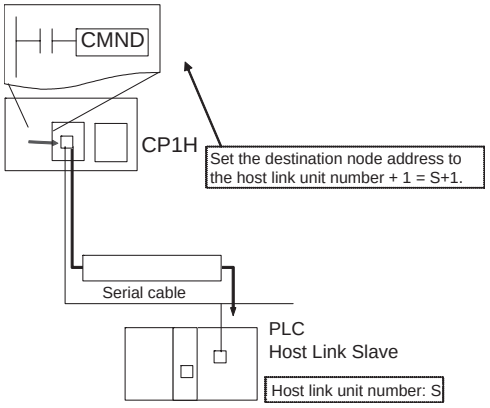
When a Serial Communications Option Board mounted on a CP1H CPU Unit or a CJ-series Serial Communications Unit is in Host Link mode and connected one-to-one with a host computer, SEND(090) can be executed to transmit data from the PLC to the host computer the next time that the PLC has the right to transmit. It is also possible to transmit to other host computers connected to other PLCs elsewhere in the network.



Sending Data to a Host Link Slave PLC Connected by Serial Gateway

If SEND(090) is used to send to a Serial Communications Option Board mounted on a CP1H CPU Unit or a CJ-series Serial Communications Unit, a command is sent from the serial port to the host computer. The command is a FINS message enclosed between a host link header and terminator. The FINS command is a MEMORY AREA WRITE command (command code 0102) and the host link header code is 0F hexadecimal.

The serial gateway function can be used to send data to a PLC connected as a Host Link Slave to a Serial Communications Board or Unit. In this case, the destination node address must be set to the host link unit number + 1.



A program must be created in the host computer to process the received command (the FINS command enclosed in the host link header and terminator).

If the destination serial port is in the local PLC, set the network address to 00 (local network) in C+1, set the node address to 00 (local PLC) in C+2, and set the unit address to 00 (Serial Communications Option Board on CPU Unit), or unit number + 10 hexadecimal (CJ-series Serial Communications Unit).

Flags

Name	Label	Operation
Error Flag	ER	ON if the serial port number specified in C+1 is not within the range of 00 to 04. ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C+3. OFF in all other cases.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of a network instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when an instruction is executed.

Precautions

If the Communications Port Enabled Flag is OFF for the port number specified in C+3, the instruction will be treated as NOP(000) and will not be executed. The Error Flag will be turned ON in this case.

When an address in the current bank of the EM Area is specified for D, the transmitted data will be written to the current EM bank of the destination node.

When data will be transmitted outside of the local network, the user must register routing tables in the PLCs (CPU Units) in each network. (Routing tables indicate the routes to other networks in which destination nodes are connected.)

Refer to the FINS command response codes in the *CS/CJ/CP Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

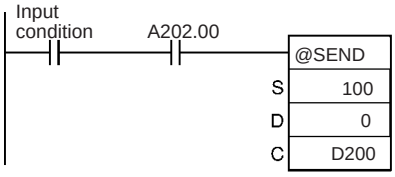
Only one network instruction may be executed for a communications port at one time. To ensure that SEND(090) is not executed while a port is busy, program the port's Communications Port Enabled Flag (A202.00 to A202.07) as a normally open condition.

Communications port numbers 00 to 07 are shared by the network instructions and PMCR(260), so SEND(090) cannot be executed simultaneously with PMCR(260) if the instructions are using the same port number.

Noise and other factors can cause the transmission or response to be corrupted or lost, so we recommend setting the number of retries to a non-zero value which will cause SEND(090) to be executed again if the response is not received within the response monitoring time.

Example 1

When the input condition and A202.00 (the Communications Port Enabled Flag for port 7) are ON in the following example, the ten words from CIO 100 to CIO 109 are transmitted to the host computer connected to port 1 of the CJ-series Serial Communications Unit with unit address 10 (hex) at node number 3 in network 0.



C	D200	0 0 0 A	Number of words to send: 10 words
C+1	D201	0 1 0 0	Transmit to network 0 and port 1 of Serial Communications Unit
C+2	D202	0 0 1 0	Node number 0, unit address 10
C+3	D203	0 0 0 0	Response requested, port number 0, no retries
C+4	D204	0 0 0 0	Response monitoring time: 2 seconds (0000: default value)

It is necessary create a program at the host computer to receive the data and send a response.

Example 2

When CIO 0.00 and A202.07 (the Communications Port Enabled Flag for port 07) are ON in the following example, the ten words from D100 to D109 are transmitted to node number 3 in the local network where they are written to the ten words from D200 to D209. The data will be retransmitted up to 3 times if a response is not received within ten seconds.

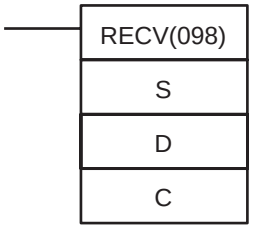
C: D300	0 0 0 A	Number of words to send: 10 words
C+1: D301	0 0 0 0	Transmit to the local network and the device itself
C+2: D302	0 3 0 0	Node number 3, unit address 00 (CPU Unit)
C+3: D303	0 7 0 3	Response requested, port number 7, 3 retries
C+4: D304	0 0 6 4	Response monitoring time: 0064 hexadecimal (10 seconds)

3-24-4 NETWORK RECEIVE: RECV(098)

Purpose

Requests data to be transmitted from a node in the network and receives the data.

Ladder Symbol



S: First source word (remote node)

D: First destination word (local node)

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	RECV(098)
	Executed Once for Upward Differentiation	@RECV(098)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: First control word

The five control words C to C+4 specify the number of words to be received, the source of the transmission, and other settings shown in the following table.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words: 0001 to maximum allowed[1] (4-digit hexadecimal)	
C+1	Source network address: 00 to 7F (0 to 127)[2, 4]	Bits 08 to 11:Serial port number (physical port) 1 hex: Port 1 2 hex: Port 2 (Do not set 0, 3, or 4.) Bits 12 to 15: Always 0.
C+2	Source unit address[5]	Source node address: 00 to maximum allowed[6]
C+3	No. of retries: 00 to 0F (0 to 15)	Port number: 00 to 07 (F: Automatic allocation)[7] Response is fixed to "required."
C+4	Response monitoring time: 0001 to FFFF (0.1 to 6553.5 seconds) (The default setting of 0000 sets a monitoring time of 2 seconds.)	

Note

(1) The maximum number of words allowed depends on the network being used. For a Controller Link, the allowed range is 0001 to 03DE (1 to 990 words).

(2) Set the source network address to 00 to specify a source within the local network. When two or more CPU Bus Units are mounted, the network address will be the unit number of the Unit with the lowest unit number.

(3) The following two methods can be used to receive data from a host computer through a serial port with the host link while initiating communications from the PLC.

(a) Set the source unit address (bits 00 to 07 of C+2) to the unit address of the CP1H CPU Unit or the Serial Communications Unit and set the serial port number (bits 08 to 11 of C+1) to 1 for port 1 or 2 for port 2.

Unit address (C+2, bits 00 to 07)	Unit	Serial port number (C+1, bits 08 to 11)	Serial port
00 hex	CP1H CPU Unit	1 hex	Port 1
		2 hex	Port 2
10 hex + unit number	CJ-series Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1
		2 hex	Port 2

(b) Set the source unit address directly into bits 00 to 07 of C+2. In this case, set the serial port number in bits 08 to 11 of C+1 to 0 for direct specification.

CPU Unit Serial Communications Option Board Ports

Port	Port's unit address
Port 1	FD hex (253 decimal)
Port 2	FC hex (252 decimal)

CJ-series Serial Communication Unit Ports

Port	Port's unit address	Example: Unit number = 1
Port 1	80 hex + 4 × unit number	80 + 4 × 1 = 84 hex (132 decimal)
Port 2	81 hex + 4 × unit number	81 + 4 × 1 = 85 hex (133 decimal)

(4) When specifying the serial port without a routing table for the serial gateway function (conversion to host link FINS), set the serial port's unit address in the source network address byte.

(5) The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
CP1H CPU Unit (Serial Communications Option Board) or CJ-series CPU Unit	00 hex
CS/CJ-series CPU Bus Unit	10 hex + unit number
CS/CJ-series Special I/O Unit	20 hex + unit number
CS-series Inner Board	E1 hex
Computer	01 hex
Unit connected to network (not necessary to specify Unit)	FE hex
Direct specification of the serial port's unit address	CP1H CPU Unit with Serial Communications Option Board Mounted Port 1: FD hex (253 decimal) Port 2: FC hex (252 decimal) CS/CJ-series Serial Communications Unit ports Port 1: 80 hex + 4 × unit number Port 2: 81 hex + 4 × unit number CS-series Serial Communications Board ports Port 1: E4 hex (228 decimal) Port 2: E5 hex (229 decimal) CS/CJ-series CPU Unit ports Peripheral port: FD hex (253 decimal) RS-232C port: FC hex (252 decimal)

(6) The maximum node number depends on the network being used. For a Controller Link, the allowed range is 00 to 20 hexadecimal (0 to 32). Set the source node number to 00 to transmit within the local node.

(7) Refer to *Automatic Allocation of Communications Ports* on page 850 for details on using automatic allocation of the communications port number (logical port).

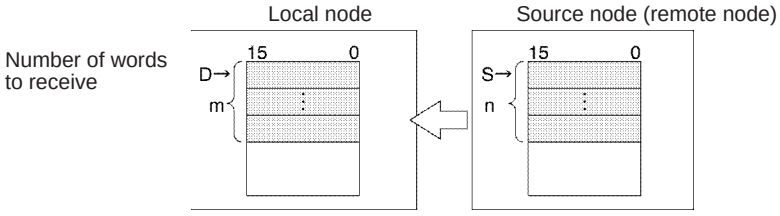
Operand Specifications

Area	S	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6139
Work Area	W0 to W511		W0 to W507
Holding Bit Area	H0 to H511		H0 to H507
Auxiliary Bit Area	A0 to A447 A448 to A959	A448 to A959	A0 to A443 A448 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D0 to D32767		D0 to D32763
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---		
Data Registers	---		

Area	S	D	C
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

RECV(098) requests the number of words specified in C beginning at word S to be transferred from the designated device to the local PLC. The data is received through the PLC's CPU Bus or over the network and written to the PLC's data area beginning at D.



A response is required with RECV(098) because the response contains the data being received. If the response has not been received within the response monitoring time set in C+4, the request for data transfer will be retransmitted up to 15 times (retries set in bits 0 to 3 of C+3).

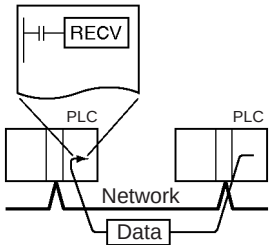
RECV(098) can be used to request a data transmission from a particular serial port in the source device as well as the device itself.

Data can be received from a host computer connected to the PLC's serial port (when set to host link mode) as well as a PLC or computer connected through a Controller Link or Ethernet network.

If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when SEND(090) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A202.00 to A202.07) and Communications Port Error Flag (ports 00 to 07: A219.00 to A219.07) will be turned OFF and 0000 will be written to the word that contains the completion code (ports 00 to 07: A203 to A210). Data will be received from the destination node once the flags have be set.

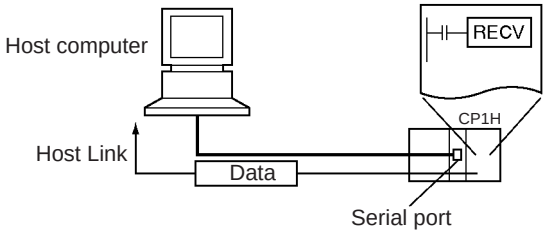
Transmission through the Network

RECV(098) can be used to receive data transmitted the specified data area in a PLC or computer connected by a Controller Link network or Ethernet link and write that data to the specified data area in the local PLC.



Transmission through Host Link

When the Serial Communications Option Board mounted on a CP1H CPU Unit or a CJ-series Serial Communications Unit is in Host Link and connected one-to-one with a host computer, RECV(098) can be executed to receive data from the host computer the next time that the PLC has the right to transmit commands. It is also possible to receive data from other host computers connected to other PLCs elsewhere in the network.



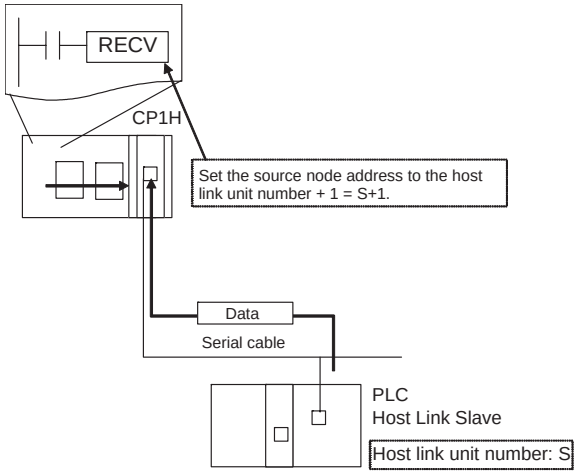
If RECV(098) is executed for the Serial Communications Option Board mounted on a CP1H CPU Unit or a CJ-series Serial Communications Unit, a command is sent from the serial port to the host computer. The command is a FINS message enclosed between a host link header and terminator. The FINS command is a MEMORY AREA READ command (command code 0101) and the host link header code is 0F hexadecimal.

A program must be created in the host computer to process the send command (the FINS command enclosed in the host link header and terminator).

If the destination serial port is in the local PLC, set the network address to 00 (local network) in C+1, set the node address to 00 (local PLC) in C+2, and set the unit address to 00 (Serial Communications Option Board on CPU Unit), or unit number + 10 hexadecimal (CJ-series Serial Communications Unit).

Receiving Data from a Host Link Slave PLC Connected by Serial Gateway

The serial gateway function can be used to receive data from a PLC connected as a host link Slave to a Serial Communications Unit. In this case, the source node address must be set to the host link unit number + 1.



Flags

Name	Label	Operation
Error Flag	ER	ON if the serial port number specified in C+1 is not within the range of 00 to 04. ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C+3. OFF in all other cases.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of a network instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Precautions

If the Communications Port Enabled Flag is OFF for the port number specified in C+3, the instruction will be treated as NOP(000) and will not be executed. The Error Flag will be turned ON in this case.

When an address in the current bank of the EM Area is specified for D, the transmitted data will be written to the current EM bank of the destination node.

When data will be transmitted outside of the local network, the user must register routing tables in the PLCs (CPU Units) in each network. (Routing tables indicate the routes to other networks in which destination nodes are connected.)

Refer to the FINS command response codes in the *CS/CJ/CP Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Only one network instruction may be executed for a communications port at one time. To ensure that RECV(098) is not executed while a port is busy, program the port's Communications Port Enabled Flag (A202.00 to A202.07) as a normally open condition.

Communications port numbers 00 to 07 are shared by the network instructions and PMCR(260), so RECV(098) cannot be executed simultaneously with PMCR(260) if the instructions are using the same port number.

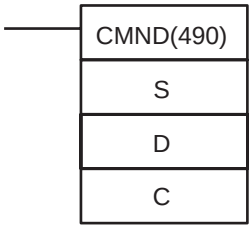
Noise and other factors can cause the transmission or response to be corrupted or lost, so we recommend setting the number of retries to a non-zero value which will cause RECV(098) to be executed again if the response is not received within the response monitoring time.

3-24-5 DELIVER COMMAND: CMND(490)

Purpose

Sends an FINS command and receives the response. Refer to the *CS/CJ/CP Series Communications Commands Reference Manual* for details on FINS commands.

Ladder Symbol



S: First command word

D: First response word

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	CMND(490)
	Executed Once for Upward Differentiation	@CMND(490)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: First control word

The six control words C to C+5 specify the number of bytes of command data and response data, the destination, and other settings shown in the following table.

Word	Bits 00 to 07	Bits 08 to 15
C	Bytes of command data: 0002 to maximum allowed[1] (4-digit hexadecimal)	
C+1	Bytes of response data: 0000 to maximum allowed[1 to 3] (4-digit hexadecimal)	
C+2	Destination network address: 00 to 7F[4, 6]	Bits 08 to 11:Serial port number (physical port) 1 hex: Port 1 2 hex: Port 2 (Do not set 0, 3, or 4.) Bits 12 to 15: Always 0.
C+3	Destination unit address: 00 to FE[5, 7, 9]	Destination node number: 00 to maximum allowed[8]
C+4	No. of retries: 00 to 0F (0 to 15)	Bits 08 to 11: Port number (internal logic port): 0 to 7 (F: Automatic allocation)[10] Bits 12 to 15: Response setting 0: Response requested. 8: No response requested.[11]
C+5	Response monitoring time: 0001 to FFFF (0.1 to 6553.5 seconds) (The default setting of 0000 sets a monitoring time of 2 seconds.)	

Note

(1) The number of bytes of command data in C is 0002 to the maximum data length in hexadecimal. For example, the number of bytes would be 0002 to 07C6 hex (2 to 1,990 bytes) for Controller Link systems. The number of bytes for the local CPU Unit is 07C6 hex (1,990 bytes). The number of bytes of command data depends on the network.

(2) The number of bytes of response data in C+1 is 0000 to the maximum data length in hexadecimal. For example, the number of bytes would be 0000 to 07C6 hex (0 to 1,990 bytes) for Controller Link systems. The number of bytes for the local CPU Unit is 07C6 hex (1,990 bytes). The number of bytes of response data depends on the network.

(3) Refer to the operation manual for the specific network for the maximum data lengths for the command data and response data. For any FINS command passing through multiple networks, the maximum data lengths for the command data and response data are determined by the network with the smallest maximum data lengths.

(4) Set the destination network address to 00 to transmit within the local network. When two or more CPU Bus Units are mounted, the network address will be the unit number of the Unit with the lowest unit number.

(5) The following two methods can be used to send a FINS command to a host computer through a serial port with the host link host link while initiating communications from the PLC, or the serial gateway function (converted to CompoWay/F, Modbus-RTU, or Modbus-ASCII).

(a) Set the destination unit address (bits 00 to 07 of C+3) to the unit address of the CP1H CPU Unit or CJ-series Serial Communications Unit and set the serial port number (bits 08 to 11 of C+2) to 1 for port 1 or 2 for port 2.

Unit address (C+3, bits 00 to 07)	Unit	Serial port number (C+2, bits 08 to 11)	Serial port
00 hex	CP1H CPU Unit	1 hex	Port 1
		2 hex	Port 2
10 hex + unit number	CJ-series Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1
		2 hex	Port 2

(b) Set the destination unit address directly into bits 00 to 07 of C+3. In this case, set the serial port number in bits 08 to 11 of C+2 to 0 for direct specification.

CPU Unit Serial Communications Option Board Ports

Port	Port's unit address
Port 1	FD hex (253 decimal)
Port 2	FC hex (252 decimal)

CJ-series Serial Communication Unit Ports

Port	Port's unit address	Example: Unit number = 1
Port 1	80 hex + 4 × unit number	80 + 4 × 1 = 84 hex (132 decimal)
Port 2	81 hex + 4 × unit number	81 + 4 × 1 = 85 hex (133 decimal)

(6) When specifying the serial port without a routing table for the serial gateway function (conversion to host link FINS), set the serial port's unit address in the destination network address byte.

(7) The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
CP1H CPU Unit (Serial Communications Option Board) or CJ-series CPU Unit	00 hex
CS/CJ-series CPU Bus Unit	10 hex + unit number
CS/CJ-series Special I/O Unit	20 hex + unit number

Unit	Unit address setting
CS-series Inner Board	E1 hex
Computer	01 hex
Unit connected to network (not necessary to specify Unit)	FE hex
Direct specification of the serial port's unit address	CP1H CPU Unit with Serial Communications Option Board Mounted Port 1: FD hex (253 decimal) Port 2: FC hex (252 decimal) CS/CJ-series Serial Communications Unit ports Port 1: 80 hex + 4 × unit number Port 2: 81 hex + 4 × unit number CS-series Serial Communications Board ports Port 1: E4 hex (228 decimal) Port 2: E5 hex (229 decimal) CS/CJ-series CPU Unit ports Peripheral port: FD hex (253 decimal) RS-232C port: FC hex (252 decimal)

(8) The maximum node number depends on the network being used. For a Controller Link, the allowed range is 00 to 20 hexadecimal (0 to 32). Set the destination node number to FF to broadcast to all nodes; set it to 00 to transmit within the local node.

(9) When specifying the serial port in the serial gateway function (conversion to host link FINS), set the destination unit address to the host link unit number of the destination PLC + 1 (setting range: 1 to 32).

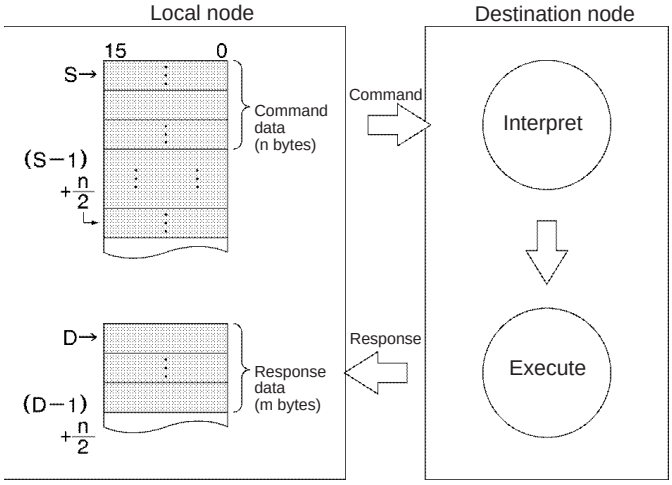
(10) Refer to *Automatic Allocation of Communications Ports* on page 850 for details on using automatic allocation of the communications port number (logical port).

(11) When the destination node number is set to FF (broadcast transmission), there will be no response even if bits 12 to 15 are set to 0.

Area	S	C	D
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6138
Work Area	W0 to W511		W0 to W506
Holding Bit Area	H0 to H511		H0 to H506
Auxiliary Bit Area	A0 to A447 A448 to A959	A448 to A959	A0 to A442 A448 to A954
Timer Area	T0000 to T4095		T0000 to T4090
Counter Area	C0000 to C4095		C0000 to C4090
DM Area	D0 to D32767		D0 to D32762
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

CMND(490) transfers the specified number of bytes of FINS command data beginning at word S to the designated device through the PLC's CPU Bus or over a network. The response is stored in memory beginning at word D.



CMND(490) can be used to transmit command data to a particular serial port in the destination device as well as the device itself. CMND(490) operates just like SEND(090) if the FINS command code is 0102 (MEMORY AREA WRITE) and just like RECV(098) if the code is 0101 (MEMORY AREA READ).

If the destination node number is set to FF, the command data will be broadcast to all of the nodes in the designated network. This is known as a broadcast transmission.

If a response is requested (bits 12 to 15 of C+4 set to 0) but a response has not been received within the response monitoring time, the command data will be retransmitted up to 15 times (retries set in bits 0 to 3 of C+3). There will be no response and no retries for broadcast transmissions. For instructions that require no response, set the response setting to "not required."

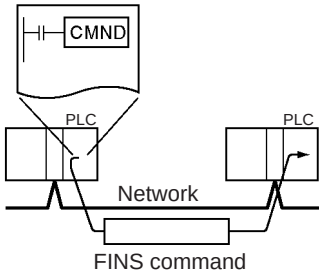
An error will occur if the amount of response data exceeds the number of bytes of response data set in C+1.

FINS command data can be transmitted to a host computer connected to a PLC serial port as well as a PLC (CP1H CPU Unit, CS/CJ-series CPU Unit, CS/CJ-series CPU Bus Unit, or CS-series Inner Board) or computer connected through a Controller Link or Ethernet network.

If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when CMND(490) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A202.00 to A202.07) and Communications Port Error Flag (ports 00 to 07: A219.00 to A219.07) will be turned OFF and 0000 will be written to the word that contains the completion code (ports 00 to 07: A203 to A210). The command data will be transmitted to the destination node(s) once the flags have be set.

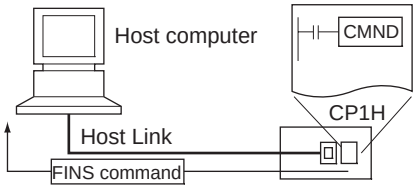
Transmission through the Network

CMND(490) can be used to transmit any FINS command to a personal computer or a PLC (CP1H CPU Unit, CS/CJ-series CPU Unit, CS/CJ-series CPU Bus Unit, or CS-series Inner Board) connected by a Controller Link network or Ethernet link.



Transmission through Host Link

When the serial port on a Serial Communications Option Board mounted to the CPU Unit's built-in serial port or a serial port on a CJ-series Serial Communications Unit is in host link mode and connected one-to-one with a host computer, CMND(490) can be executed to transmit any FINS command from the PLC to the host computer the next time that the PLC has the right to transmit. It is also possible to transmit to other host computers connected to other PLCs elsewhere in the network.



CMND(490) can be executed for the either port on the CPU Unit (Serial Communications Option Boards) or CJ-series Serial Communications Unit to send a command to the connected host computer. (Specify the serial port as 1 hex or 2 hex in bits 08 to 11 of C+2.) The command is a FINS message enclosed between a host link header and terminator. Any FINS command command can be sent; the host link header code is 0F hexadecimal.

A program must be created in the host computer to process the received command (the FINS command enclosed in the host link header and terminator).

If the destination serial port is in the local PLC, set the network address to 00 (local network) in C+2, set the node address to 00 (local PLC) in C+3, and set the unit address to 00 (Serial Communications Option Boards on a CP1H CPU Unit), or unit number + 10 hexadecimal (CJ-series Serial Communications Unit).

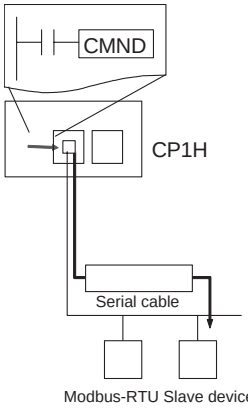
Serial Gateway Communications to a Component or Host Link Slave

It is possible to send FINS commands (or send/receive data) to a component or Host Link Slave connected to the PLC through a serial port on a CP1H Serial Communications Option Board or CJ-series Serial Communications Unit using the serial gateway function.

- Sending to a Component
  (Conversion to CompoWay/F, Modbus-RTU, or Modbus-ASCII)

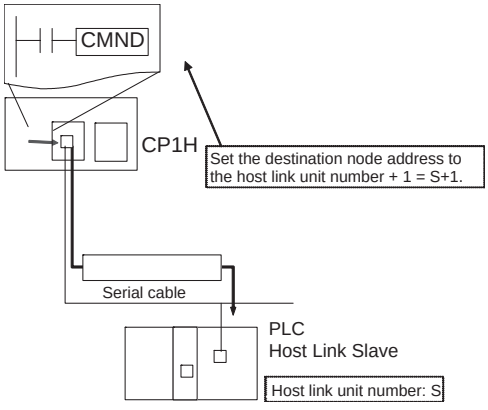
  The serial gateway function can convert the following FINS commands to CompoWay/F, Modbus-RTU, or Modbus-ASCII commands when the FINS command is sent to a Serial Communications Board or Unit's serial port or one of the CPU Unit's serial ports (peripheral or RS-232C).

  Convert to CompoWay/F command: 2803 hex
  Convert to Modbus-RTU command: 2804 hex
  Convert to Modbus-ASCII command: 2805 hex (Not supported for serial ports on CP1H Serial Communications Option Boards.)



- Sending to a PLC operating as a Host Link Slave

  The serial gateway function can be used to send any FINS command to a PLC that is connected as a host link slave and through a serial port on a CP1H Serial Communications Option Board or CJ-series Serial Communications Unit. In this case, the destination node address must be set to the host link unit number + 1.



Flags

Name	Label	Operation
Error Flag	ER	ON if the serial port number specified in C+2 is not within the range of 00 to 04. ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C+4. OFF in all other cases.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of a network instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Precautions

If the Communications Port Enabled Flag is OFF for the port number specified in C+4, the instruction will be treated as NOP(000) and will not be executed. The Error Flag will be turned ON in this case.

When data will be transmitted outside of the local network, the user must register routing tables in the PLCs (CPU Units) in each network. (Routing tables indicate the routes to other networks in which destination nodes are connected.)

Refer to the FINS command response codes in the *CS/CJ/CP Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Communications port numbers 00 to 07 are shared by the network and serial communications instructions (SEND(090), RECV(098), CMND(490), PMCR(260), TXDU(256), or RXDU(255)), so only one of these instructions may be executed for a communications port at one time. To ensure that CMND(490) is not executed while a port is busy, program the port's Communications Port Enabled Flag (A202.00 to A202.07) as a normally open condition.

Noise and other factors can cause the transmission or response to be corrupted or lost, so we recommend setting the number of retries to a non-zero value which will cause CMND(490) to be executed again if the response is not received within the response monitoring time.

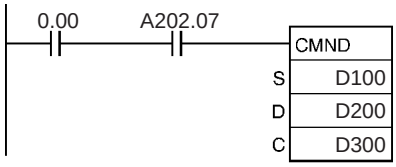
Examples

The following program section shows an example of sending a FINS command to another CPU Unit.

When CIO 0.00 and A202.07 (the Communications Port Enabled Flag for port 07) are ON, CMND(490) transmits FINS command 0101 (MEMORY AREA READ) to node number 3. The response is stored in D200 to D211.

The MEMORY AREA READ command reads 10 words from D10 to D19. The response contains the 2-byte command code (0101), the 2-byte completion code, and then the 10 words of data, for a total of 12 words or 24 bytes.

The data will be retransmitted up to 3 times if a response is not received within ten seconds.



	15 87 0	
S: D100	0 1 : 0 1	Command code: 0101 hexadecimal (MEMORY AREA READ)
S+1: D101	8 2 : 0 0	D10 (Data area = 82 hexadecimal, address = 000A00)
S+2: D102	0 A : 0 0	
S+3: D103	0 0 : 0 A	Number of words to read = 0A hexadecimal (10 decimal)

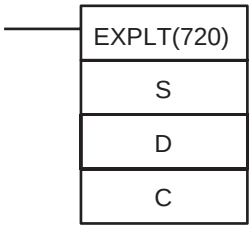
	15 87 0	
C: D300	0 0 : 0 8	Bytes of command data: 0008 (8 decimal)
C+1: D301	0 0 : 1 8	Bytes of response data: 0018 (24)
C+2: D302	0 0 : 0 0	Transmit to the local network and the device itself
C+3: D303	0 3 : 0 0	Node number 3, unit address 00 (CPU Unit)
C+4: D304	0 7 : 0 3	Response requested, port number 7, 3 retries
C+5: D305	0 0 : 6 4	Response monitoring time: 0064 hexadecimal (10 seconds)

3-24-6 EXPLICIT MESSAGE SEND: EXPLT(720)

Purpose

Sends an explicit message with any service code.

Ladder Symbol



S: First word of send message

D: First word of received message

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	EXPLT(720)
	Executed Once for Upward Differentiation	@EXPLT(720)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

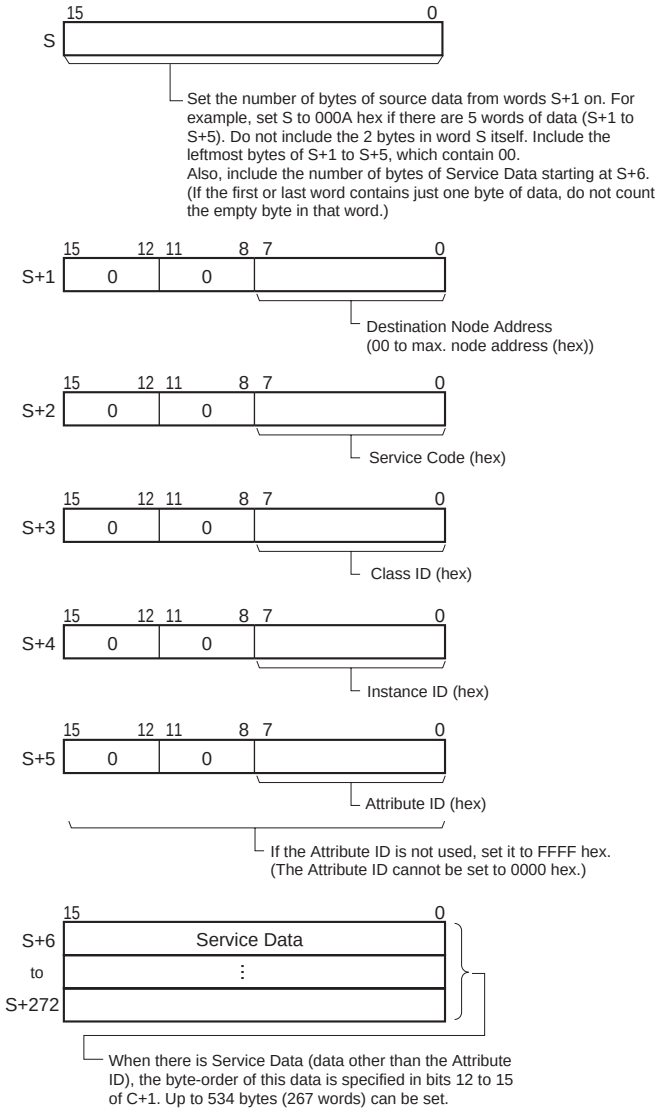
Applicable Program Areas

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

Operands

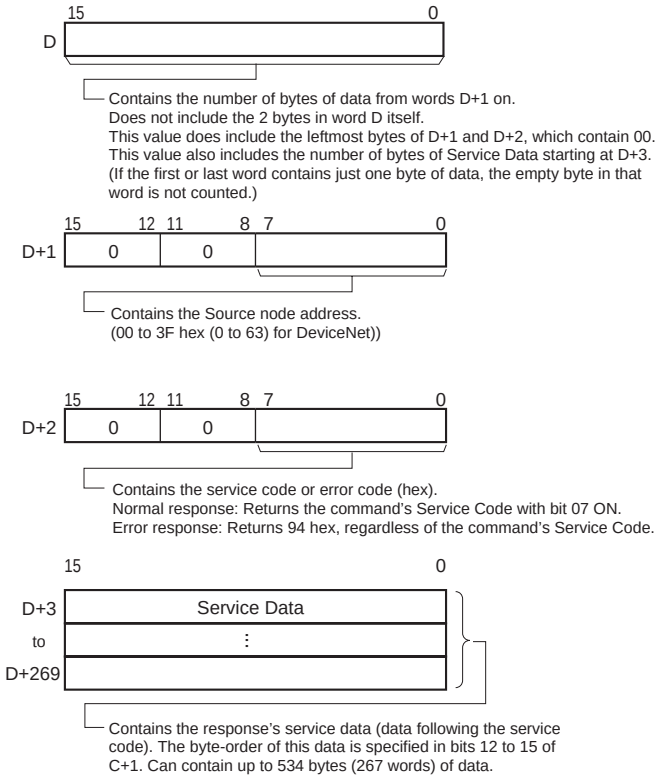
S: First word of send message

Specifies the first word of the send message (S to S+272 max.).



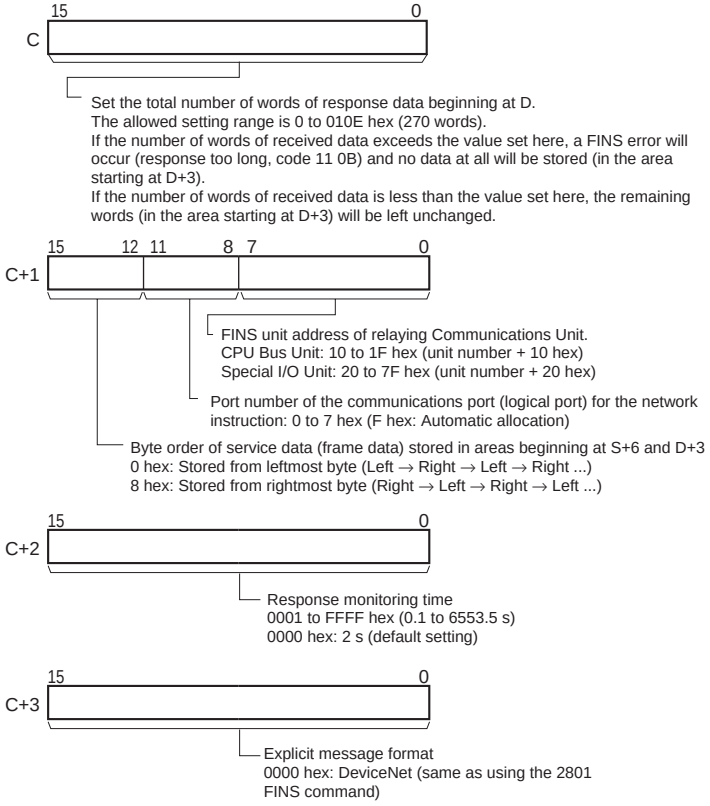
D: First word of received message

Specifies the first word of the received message (D to D+269 max.).



C: First control word

Specifies the first of four control words (C to C+3).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6140
Work Area	W0 to W511		W0 to W508
Holding Bit Area	H0 to H511		H0 to H508
Auxiliary Bit Area	A0 to A959	A448 to A959	A0 to A956
Timer Area	T0000 to T4095		T0000 to T4092
Counter Area	C0000 to C4095		C0000 to C4092
DM Area	D0 to D32767		D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

Sends the explicit message command (stored in the range of words beginning at S+2) to the node address specified in S+1, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C+1. When the response to the explicit message is received, it is stored in the range of words beginning at D+2.

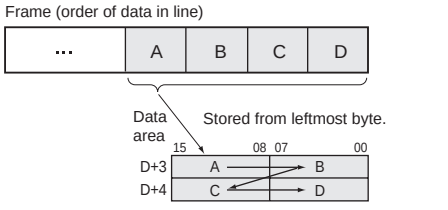
Number of Bytes Settings

The number of bytes of send data in S includes the 10 bytes in S+1 to S+5 as well as the number of bytes of service data beginning at S+6. (For example, if there is 1 byte of service data, there are 11 bytes of data all together, so S must be set to 000B hex.)

The number of bytes of received data in D includes the 4 bytes in D+1 and D+2 as well as the number of bytes of service data beginning at D+3. (For example, if there is 1 byte of service data, there are 5 bytes of data all together and D contains 0005 hex.)

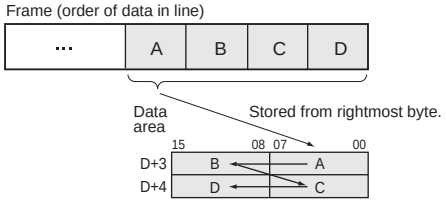
The setting in bits 12 to 15 of C+1 (0 or 8 hex) determines the byte-order of the service data stored at S+6 and D+3.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C+1 to 0 hex.



Note: A, B, C, and D represent bytes of data.

- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

For details on the general operation of the explicit message instructions, refer to *3-24-2 About Explicit Message Instructions*.

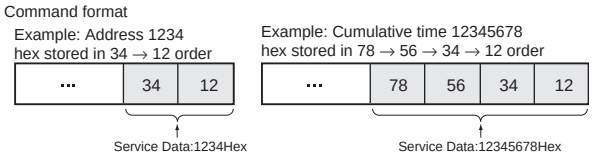
The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Explicit Communications Error Flag	A213.00 to A213.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.

Name	Address	Operation
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

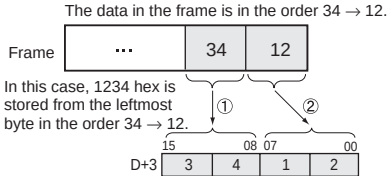
Precautions

Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.

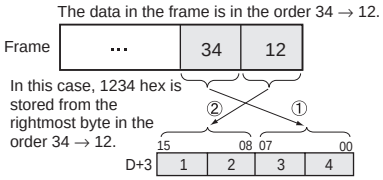


The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

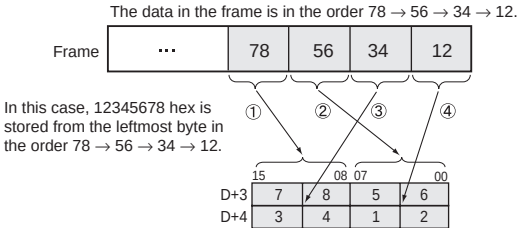
1. Data in 2-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 1234 hex in D+3



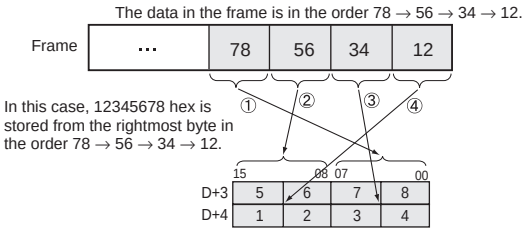
- Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
  Example: Storing the value 1234 hex in D+3



2. Data in 4-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 12345678 hex in D+3 and D+4



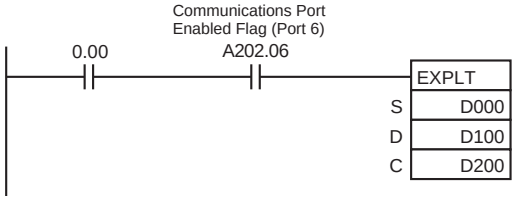
   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 12345678 hex in D+3 and D+4



Note The examples above only show the storage of received data in D+3, but send data is stored in S+6 in the same way.

Example

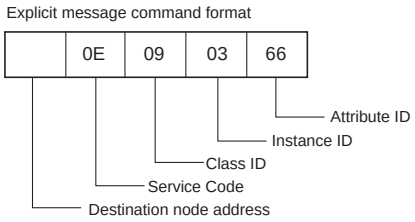
In this example, EXPLT(720) is used to read the total ON time or number of contact operations from a DRT2 Slave (I/O Terminal).

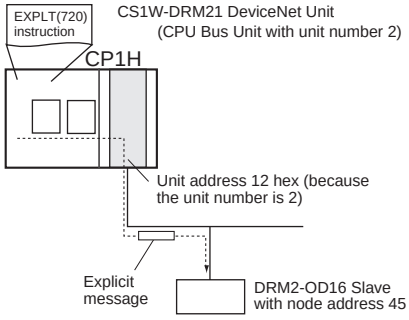


When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, EXPLT(720) reads the Total ON Time (s) or Number of Contact Operations from a DRT2 Slave (I/O Terminal). In this case, the Total ON Time or Number of Contact Operations for input 3 are read.

Service Code = 0E hex, Class ID = 09 hex, Instance ID = 03 hex, and Attribute ID = 66 hex.

For example, a value of 2,752,039 s is returned as the response for the Total ON Time.





S:	D0	0	0	0	A	Number of bytes of data: S+1 to S+5 = 5 words = 10 bytes = 0A hex
S+1:	D1	0	0	2	D	Slave's node address = 45 = 2D hex
S+2:	D2	0	0	0	E	Service Code = 0E hex
S+3:	D3	0	0	0	9	Class ID = 09 hex
S+4:	D4	0	0	0	3	Instance ID = 03 hex (Input 3)
S+5:	D5	0	0	6	6	Attribute ID = 66 hex

D:	D100	0	0	0	8	Contains 08 hex for 8 bytes of received data in response frame.
D+1:	D101	0	0	2	D	Returns Slave's node address = 45 = 2D hex.
D+2:	D102	0	0	8	E	Service Code = 8E hex (normal completion)
D+3:	D103	2	7	F	E	Service Data = 0029FE27 hex (2,752,039 s decimal)
D+4:	D104	2	9	0	0	

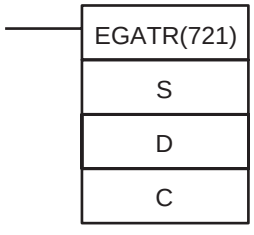
C:	D200	0	0	0	4	Set 5 words = 0005 hex since there are 5 words in D to D+5.
C+1:	D201	0	6	1	2	Byte order = 0 hex (from leftmost byte), communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+2:	D202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+3:	D203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

3-24-7 EXPLICIT GET ATTRIBUTE: EGATR(721)

Purpose

Sends an information/status read command in an explicit message (Get Attribute Single, ServiceCode: 0E hex).

Ladder Symbol



S: First word of send message

D: First word of received message

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	EGATR(721)
	Executed Once for Upward Differentiation	@EGATR(721)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

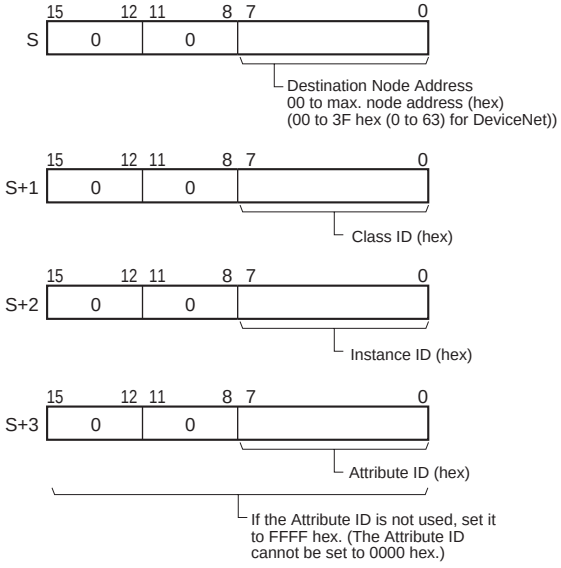
Applicable Program Areas

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

Operands

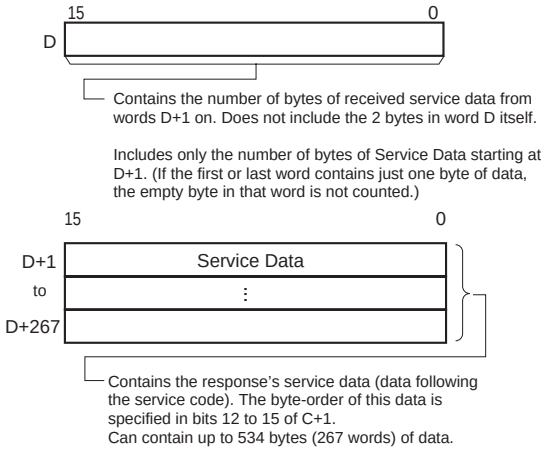
S: First word of send message

Specifies the first word of the send message (S to S+3).



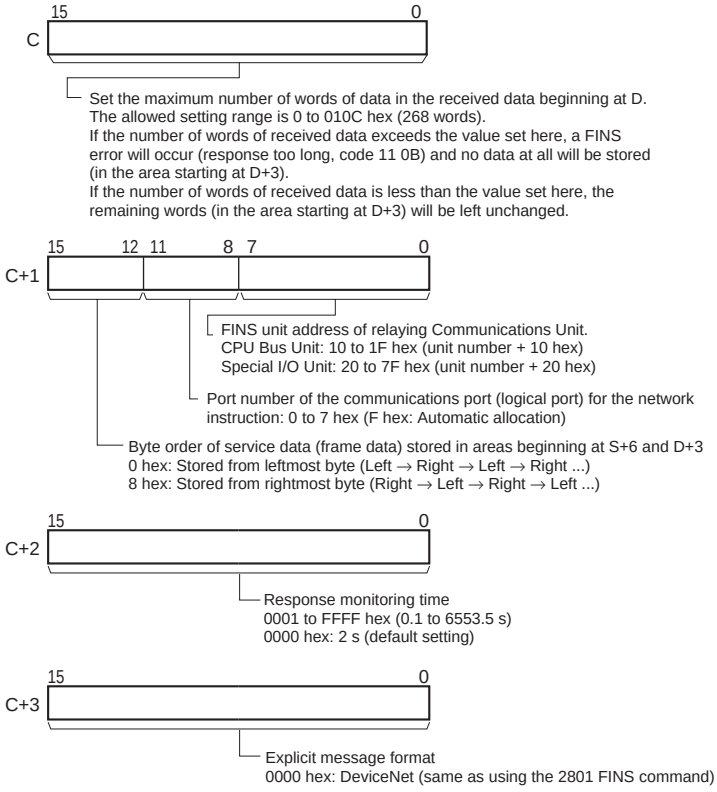
D: First word of received message

Specifies the first word of the received message (D to D+267 max.).



C: First control word

Specifies the first of four control words (C to C+3).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0 to CIO 6140	CIO 0 to CIO 6143	CIO 0 to CIO 6140
Work Area	W0 to W508	W0 to W511	W0 to W508
Holding Bit Area	H0 to H508	H0 to H511	H0 to H508
Auxiliary Bit Area	A0 to A956	A0 to A959	A0 to A956
Timer Area	T0000 to T4092	T0000 to T4095	T0000 to T4092
Counter Area	C0000 to C4092	C0000 to C4095	C0000 to C4092
DM Area	D0 to D32764	D0 to D32767	D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

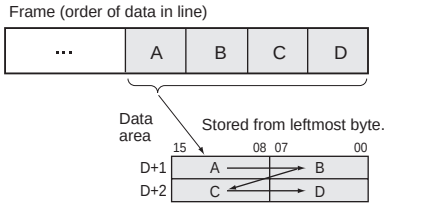
Description

Sends the “read information/status” explicit message command (stored in words S+1 to S+3) to the node address specified in S, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C+1.

When the response to the explicit message is received, the response's service data (data following the service code) is stored in the range of words beginning at D+1.

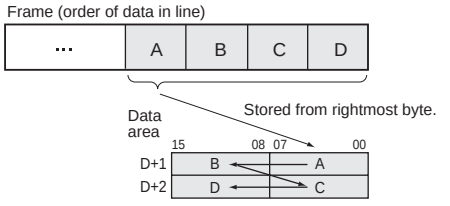
The number of bytes of received data indicated in D is the number of bytes of service data. (For example, if there is 1 byte of service data, D will contains 0001 hex. D will contain 0001 hex regardless of the byte order setting, i.e., whether the byte is stored in the rightmost or leftmost byte of D.)
The setting in bits 12 to 15 of C+1 (0 or 8 hex) determines the byte-order of the service data stored at S+6 and D+3.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C+1 to 0 hex.



Note: A, B, C, and D represent bytes of data.

- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

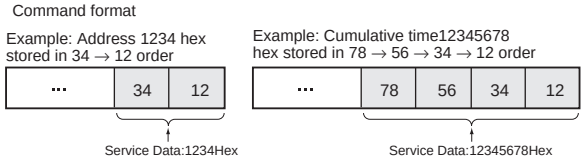
For details on the general operation of the explicit message instructions, refer to *3-24-2 About Explicit Message Instructions*.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Explicit Communications Error Flag	A213.00 to A213.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

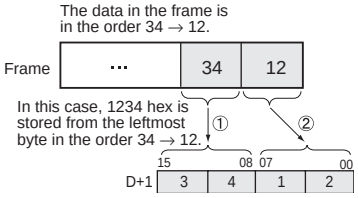
Precautions

Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.

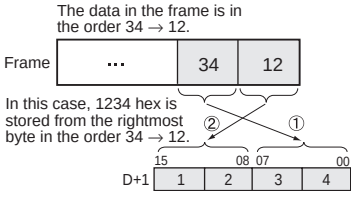


The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

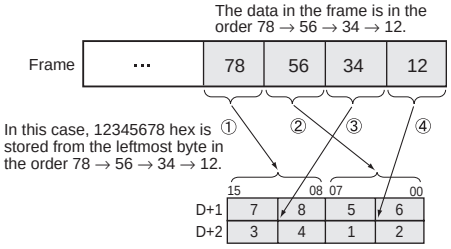
1. Data in 2-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 1234 hex in D+1



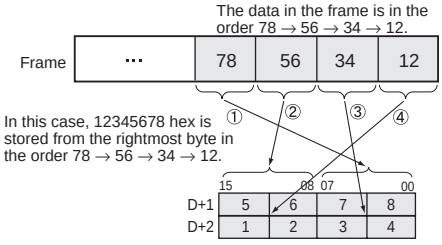
   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 1234 hex in D+1



2. Data in 4-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 12345678 hex in D+1 and D+2

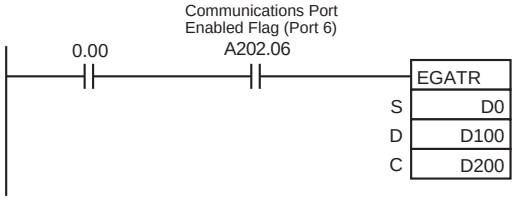


   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 12345678 hex in D+1 and D+2



Example

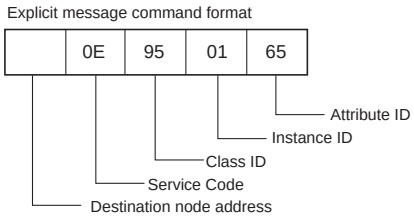
In this example, EGATR(721) is used to read the general status of a DRT2 Slave (I/O Terminal).

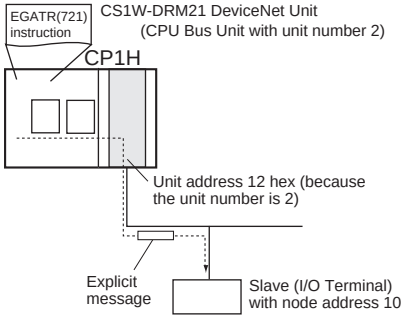


When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, EGATR(721) reads the general status of the DRT2 Slave (I/O Terminal). In this case, the Total ON Time or Number of Contact Operations for input 3 are read.

Service Code = 0E hex, Class ID = 95 hex, Instance ID = 01 hex, and Attribute ID = 65 hex.

The general status is returned in 1 byte.

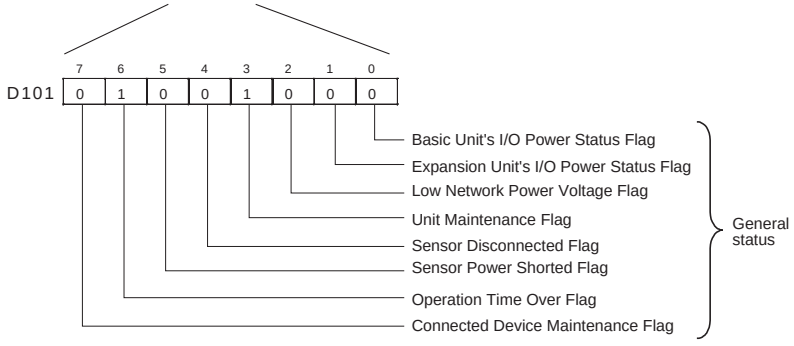




S:	D0	0	0	0	A	Slave's node address = 10 = 0A hex
S+1:	D1	0	0	9	5	Class ID = 95 hex
S+2:	D2	0	0	0	1	Instance ID = 01 hex
S+3:	D3	0	0	6	5	Attribute ID = 65 hex

C:	D200	0	0	0	2	Set 2 words = 0002 hex since there are 2 words in D to D+1.
C+1:	D201	8	6	1	2	Byte order = 8 hex (from rightmost byte), communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+2:	D202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+3:	D203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

D:	D100	0	0	0	1	D contains 0 hex for the 1 byte of data returned to the rightmost byte of D+1. The Slave's general status is returned to bits 00 to 07. (The data is stored in bits 00 to 07 because the byte order setting in C+1 bits 12 to 15 was set to 8 hex (from rightmost byte).
D+1:	D101	0	0	4	8	

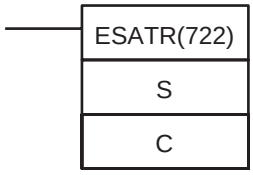


3-24-8 EXPLICIT SET ATTRIBUTE: ESATR(722)

Purpose

Sends an information write command in an explicit message (Set Attribute Single, ServiceCode: 10 hex).

Ladder Symbol



S: First word of send message

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	ESATR(722)
	Executed Once for Upward Differentiation	@ESATR(722)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

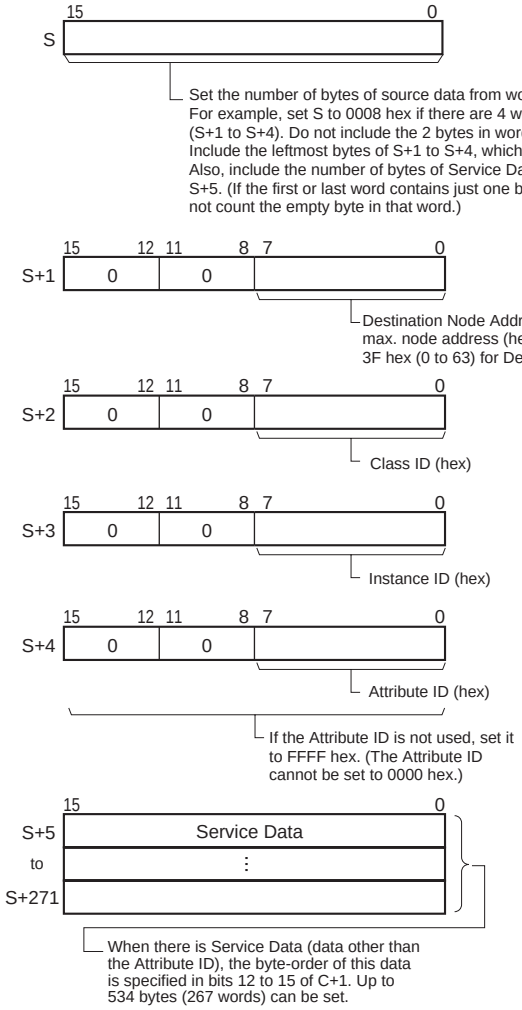
Applicable Program Areas

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

Operands

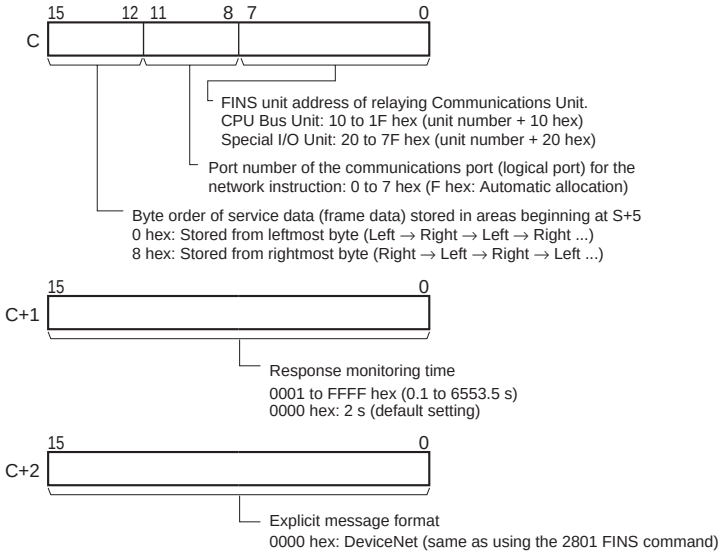
S: First word of send message

Specifies the first word of the send message (S to S+271 max.).



C: First control word

Specifies the first of three control words (C to C+2).



Operand Specifications

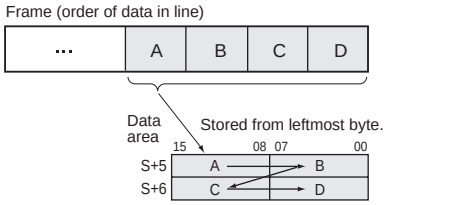
Area	S	C
CIO Area	CIO 0 to CIO 6143	CIO 0 to CIO 6141
Work Area	W0 to W511	W0 to W509
Holding Bit Area	H0 to H511	H0 to H509
Auxiliary Bit Area	A0 to A959	A0 to A957
Timer Area	T0000 to T4095	T0000 to T4093
Counter Area	C0000 to C4095	C0000 to C4093
DM Area	D0 to D32767	D0 to D32765
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

Sends the explicit message command with service code 10 hex (stored in the range of words beginning at S+2) to the node address specified in S+1, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C. When the response to the explicit message is received, it is stored in the range of words beginning at D+2.

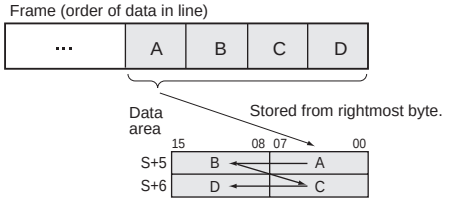
The setting in bits 12 to 15 of C (0 or 8 hex) determines the byte-order of the service data stored at S+5.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C to 0 hex.



Note: A, B, C, and D represent bytes of data.

- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C to 8 hex.



Note: A, B, C, and D represent bytes of data.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

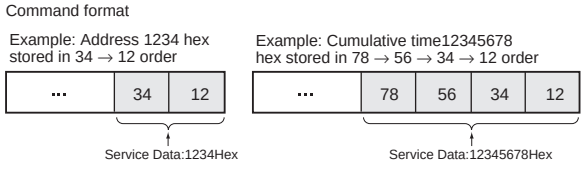
For details on the general operation of the explicit message instructions, refer to *3-24-2 About Explicit Message Instructions*.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Explicit Communications Error Flag	A213.00 to A213.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Precautions

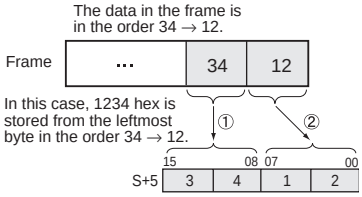
Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.



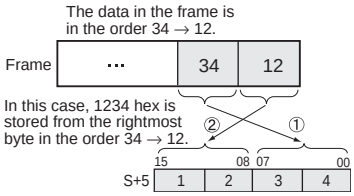
The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

1. Data in 2-byte Units

   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 1234 hex in S+5

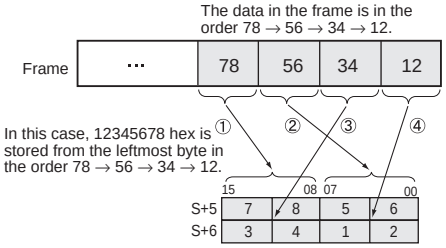


   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 1234 hex in S+5

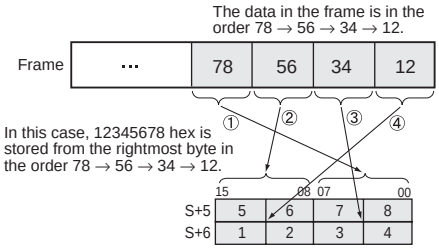


2. Data in 4-byte Units

   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 12345678 hex in S+5 and S+6

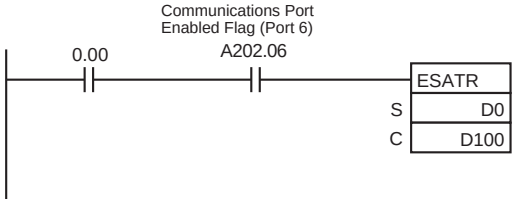


   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 12345678 hex in S+5 and S+6



Example

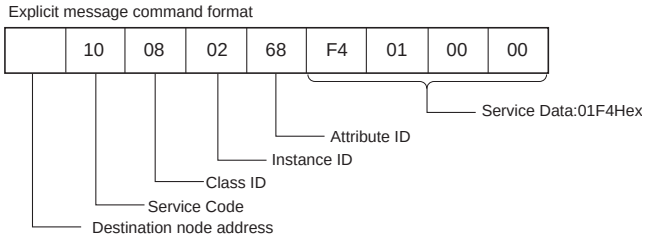
In this example, ESATR(722) is used to overwrite the Number of Contact Operations set value in a DRT2 Slave (I/O Terminal).

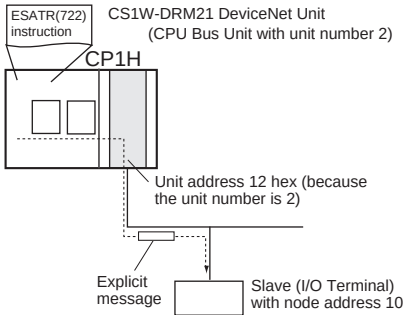


When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, EXPLT(720) writes the Number of Contact Operations set value for input 2 in a DRT2 Slave (I/O Terminal).

(Service Code = 10 hex,) Class ID = 08 hex, Instance ID = 02 hex, and Attribute ID = 68 hex.

In this case, the Number of Contact Operations is being set to 500 (1F4 hex), so the service data is set to 000001F4.





S	D0	0	0	0	C	Number of bytes of data: S+1 to S+6 = 6 words = 12 bytes = 0C hex
S:+1	D1	0	0	0	A	Slave's node address = 10 = 0A hex
S+2:	D2	0	0	0	8	Class ID = 08 hex
S+3:	D3	0	0	0	2	Instance ID = 02 hex
S+4:	D4	0	0	6	8	Attribute ID = 68 hex
S+5:	D5	0	1	F	4	Service Data = F401 hex
S+6:	D6	0	0	0	0	

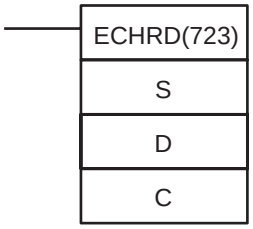
C:	D201	8	6	1	2	Byte order = 8 hex (from rightmost byte), communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+1:	D202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+2:	D203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

3-24-9 EXPLICIT WORD READ: ECHRD(723)

Purpose

Reads data to the local CPU Unit from another CPU Unit in the network.

Ladder Symbol



S: First source word in remote CPU Unit

D: First destination word in local CPU Unit

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	ECHRD(723)
	Executed Once for Upward Differentiation	@ECHRD(723)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word in Remote CPU Unit

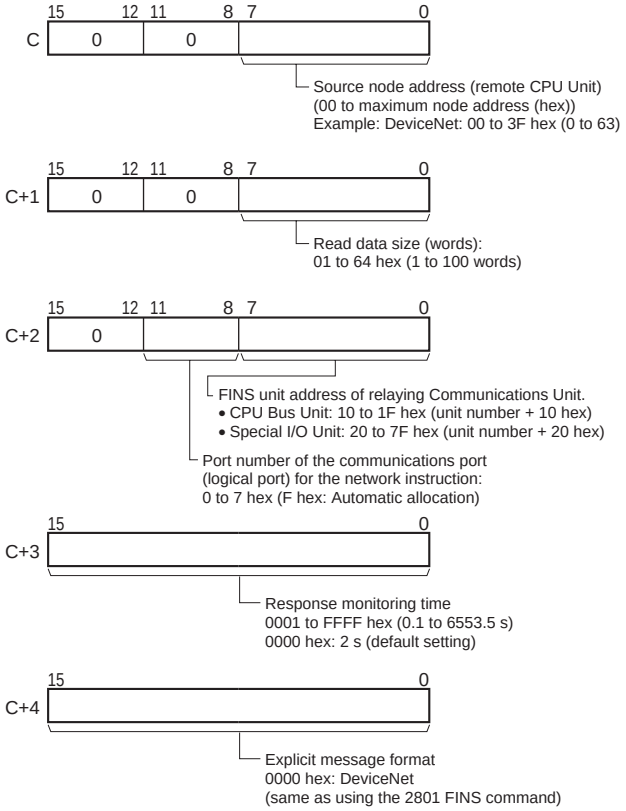
Specifies the leading word address containing the data to be read from the remote CPU Unit.

D: First Destination Word in Local CPU Unit

Specifies the leading word address where the read data will be stored in the local CPU Unit.

C: First Control Word

Specifies the first of five control words (C to C+4).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6139
Work Area	W0 to W511		W0 to W507
Holding Bit Area	H0 to H511		H0 to H507
Auxiliary Bit Area	A0 to A959	A448 to A959	A0 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091

Area	S	D	C
DM Area	D0 to D32767		D0 to D32763
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

Reads the specified number of words from the first read word (specified in S) in the remote CPU Unit with the node address specified in C, and stores the data in the local CPU Unit memory words beginning at D.

Note ECHRD(723) sends an explicit message with the Service Code 1C hex (Byte Data Read).

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

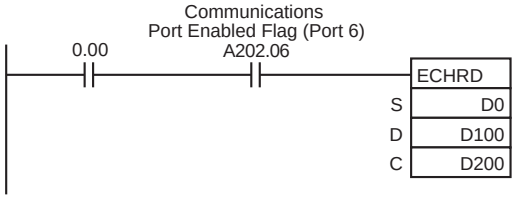
For details on the general operation of the network instructions, refer to *3-24-2 About Explicit Message Instructions*.

The following table shows relevant bits and flags in the Auxiliary Area.

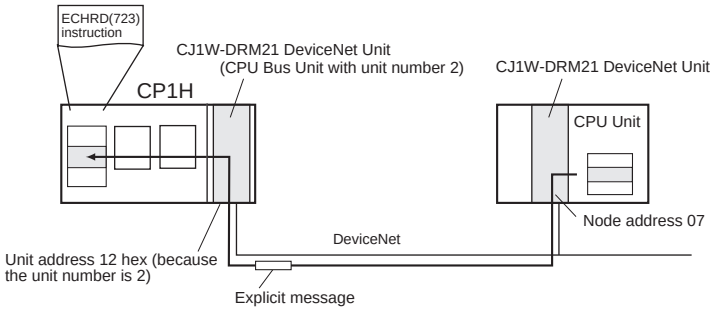
Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Explicit Communications Error Flag	A213.00 to A213.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

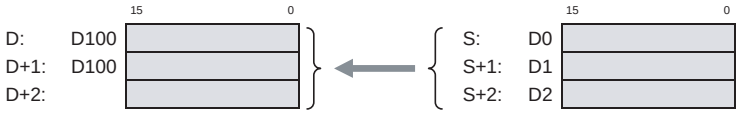
Example

In this example, ECHRD(723) is used to read the I/O memory of a CJ-series CPU Unit on the DeviceNet network, and store the data in the I/O memory of the local CPU Unit.



When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 6) are ON, ECHRD(723) reads D0 to D2 from the I/O memory of the CJ-series CPU Unit with node address 7 on the DeviceNet Network and stores the data in D100 to D102 of the local CPU Unit.





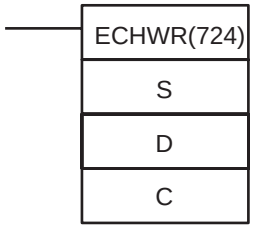
		15	8	7	0	
C:	D200	0	0	0	7	Node address of remote CPU Unit to be read = 07 hex (node 07)
C+1:	D201	0	0	0	3	Read data size (number of words) = 3 hex
C+2:	D202	0	6	1	2	Communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+3:	D203	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+4:	D204	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

3-24-10 EXPLICIT WORD WRITE: ECHWR(724)

Purpose

Writes data from the local CPU Unit to another CPU Unit in the network.

Ladder Symbol



S: First source word in local CPU Unit

D: First destination word in remote CPU Unit

C: First control word

Variations

Variations	**Executed Each Cycle for ON Condition**	ECHWR(724)
	Executed Once for Upward Differentiation	@ECHWR(724)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word in Local CPU Unit

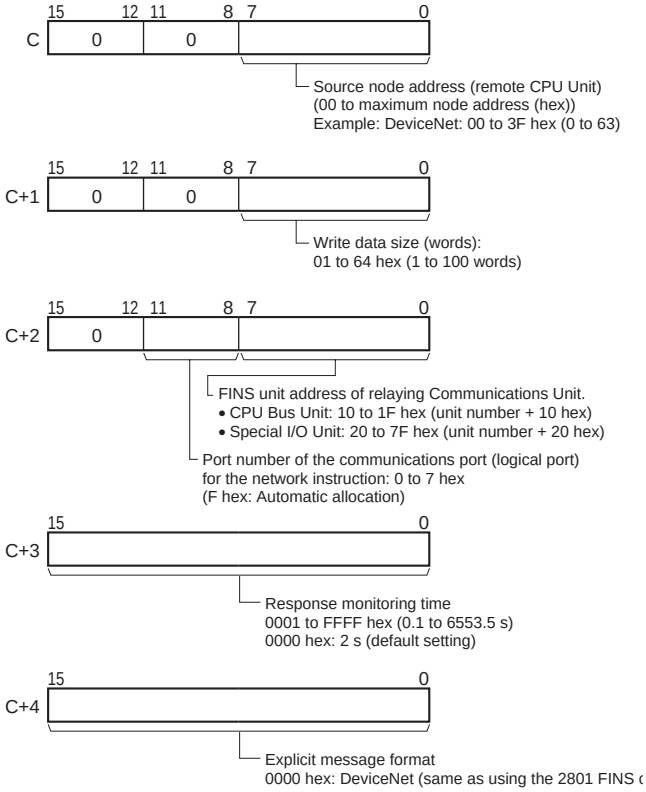
Specifies the leading word address in the local CPU Unit containing the write data.

D: First Destination Word in Remote CPU Unit

Specifies the leading word address of the write destination in the remote CPU Unit.

C: First Control Word

Specifies the first of five control words (C to C+4).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0 to CIO 6143		CIO 0 to CIO 6139
Work Area	W0 to W511		W0 to W507
Holding Bit Area	H0 to H511		H0 to H507
Auxiliary Bit Area	A0 to A959	A448 to A959	A0 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D0 to D32767		D0 to D32763
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

Writes the specified number of words beginning at S from the local CPU Unit to the write destination beginning at D in the remote CPU Unit with the node address specified in C.

Note ECHWR(724) sends an explicit message with the Service Code 1E hex (Byte Data Write).

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

For details on the general operation of the explicit message instructions, refer to *3-24-2 About Explicit Message Instructions*.

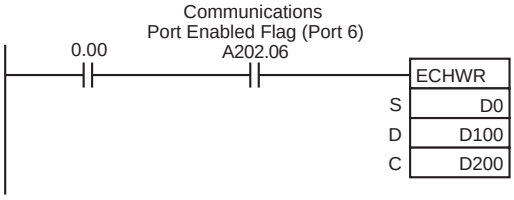
The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Explicit Communications Error Flag	A213.00 to A213.07	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.

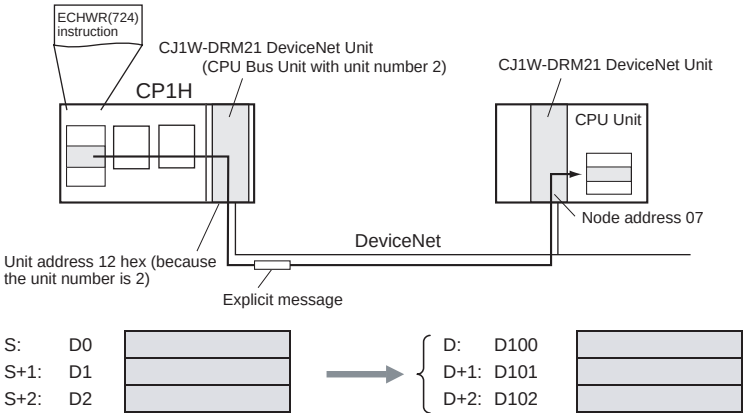
Name	Address	Operation
Communications Port Error Flag	A219.00 to A219.07	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Example

In this example, ECHWR(724) is used to write data from the I/O memory of the local CPU Unit to the I/O memory of a CJ-series CPU Unit on the DeviceNet Network.



When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 6) are ON, ECHWR(724) reads D0 to D2 from the I/O memory of the local CPU Unit and stores the data in D100 to D102 of the CJ-series CPU Unit with node address 7 on the DeviceNet Network.



		15	8	7	0	
C:	D200	0	0	0	7	Node address of remote CPU Unit to be written to = 07 hex (node 07)
C+1:	D201	0	0	0	3	Write data size (number of words) = 3 hex
C+2:	D202	0	6	1	2	Communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+3:	D203	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+4:	D204	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

3-25 Display Instructions

This section describes instructions used to display messages.

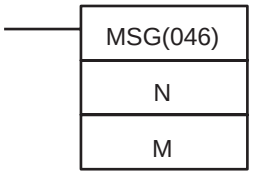
Instruction	Mnemonic	Function code	Page
DISPLAY MESSAGE	MSG	046	909
SEVEN-SEGMENT LED WORD DATA DISPLAY	SCH	047	911
SEVEN-SEGMENT LED CONTROL	SCTRL	048	913

3-25-1 DISPLAY MESSAGE: MSG(046)

Purpose

Reads the specified sixteen words of extended ASCII and displays the message on the Programming Device.

Ladder Symbol



N: Message number

M: First message word

Variations

Variations		
	Executed Each Cycle for ON Condition	MSG(046)
	Executed Once for Upward Differentiation	@MSG(046)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Message number

The message number must be 0000 to 0007 hexadecimal (or 0 to 7 decimal).

M: First message word

When displaying a message, M specifies the address of the first of the words containing the ASCII message. When clearing a message, M can be any hexadecimal constant (0000 through FFFF).

Operand Specifications

Area	M	N
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	

Area	M	N
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000 to #0007 (binary) or &0 to &7	#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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

When the execution condition is ON, MSG(046) registers the 16 words of ASCII data (up to 32 characters including the null character) from M to M+15 for the message number specified by N. Once a message has been registered, the Programming Device can be connected and the message will be displayed after any error messages that have been generated.

After a message has been registered, the message display can be changed by overwriting the message in the message storage area.

To clear a message that has been registered, execute MSG(046) with S set to the message number of the message you want to clear and N set to a constant (0000 to FFFF).

A message registered during program execution will be retained even if program execution is stopped, but all messages will be cleared when the program is executed again.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of S is not 0000 to 0007 hexadecimal. OFF in all other cases.

Precautions

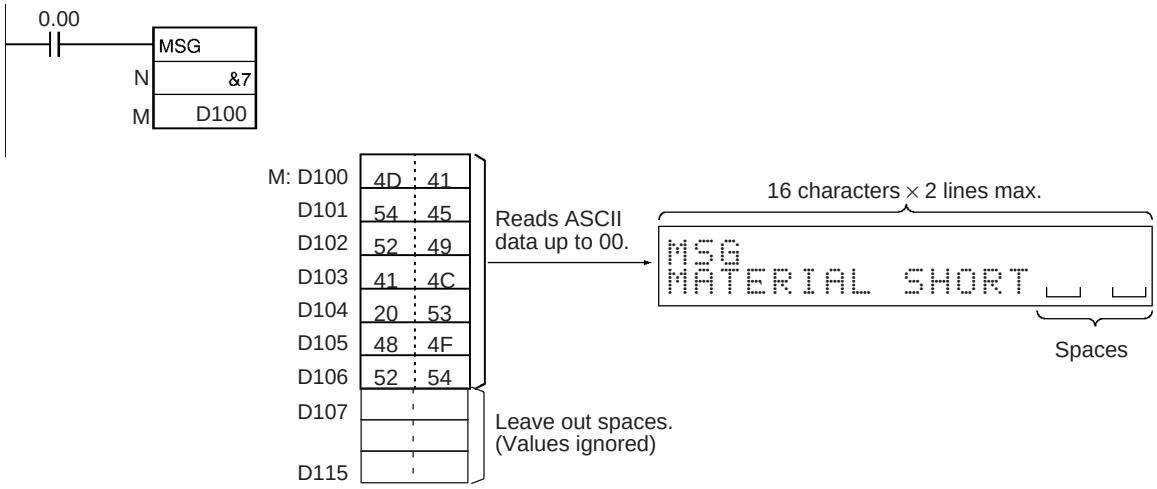
Registered messages are updated each time MSG(046) is executed.

All message characters after the null character (00) are converted to spaces in the Programming Device display.

The character stored in the leftmost byte is displayed before the character in the rightmost byte.

Examples

When CIO 0.00 turns ON in the following example, the 16 words of data in D100 through D115 are read as the 32 characters of ASCII data for message number 7 and displayed at the Programming Device.



ASCII

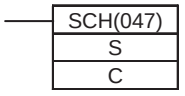
Four rightmost bits \ Four leftmost bits	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p				―	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			”	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			’	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	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	°		

3-25-2 SEVEN-SEGMENT LED WORD DATA DISPLAY: SCH(047)

Purpose

SCH(047) displays two user-specified digits on the 7-segment display on the CPU Unit.

Ladder Symbol



S: Display source word
C: Control word (digit specification)

Variations

Variations		
	Executed Each Cycle for ON Condition	SCH(047)
	Executed Once for Upward Differentiation	@SCH(047)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: Display source word

C: Control word (digit specification)

0000 hex: Display leftmost two digits
0001 hex: Display rightmost two digits

Operand Specifications

Area	S	C
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	#0000 to #FFFF	#0000, #0001
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

When the execution condition goes ON, SCH(047) displays the leftmost or rightmost two hexadecimal digits of S on the 7-segment display on the CPU Unit. Any value between 00 and FF hexadecimal can be displayed. The value of C determines if the leftmost two digits or rightmost two digits are displayed.

The display will remain even if the execution condition of SCH(047) goes OFF. Use SCTRL(048) to clear the display.

If the contents of C is not 0000 or 0001 hex, an error will occur, the ER Flag will turn ON, and nothing will be displayed.

Flags

Name	Label	Operation
Error Flag	ER	C contains a value other than 0000 or 0001 hex. OFF in all other cases.

Precautions

If displays are created by more than one SCTRL(048) or SCH(047) instruction, the last one that is executed with take priority.

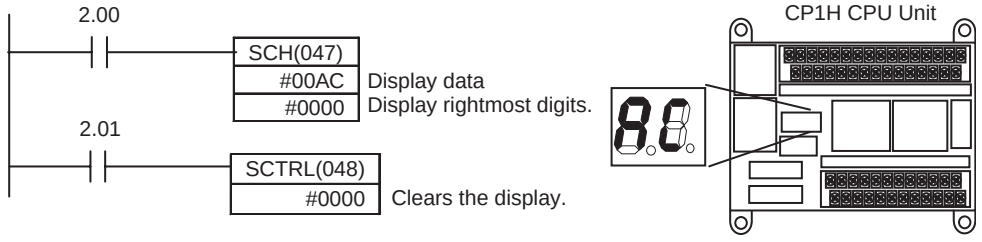
Messages on the 7-segment display on the CPU Unit are displayed in the following order of priority. Therefore, system error message will be given priority even if there is a display created with SCTRL(048) or SCH(047).

1. Error messages
2. SCTRL(048) or SCH(047) displays
3. Analog adjustment displays
4. Memory Cassette processing displays

If an error occurs while a SCTRL(048) or SCH(047) display is in effect, the error message will be given priority, but the SCTRL(048) or SCH(047) display will return when the error is cleared.

Example

When CIO 2.00 turns ON in the following example, "AC" is displayed on the 7-segment display on the CPU Unit. The display remains even when CIO 2.00 turns OFF. When CIO 2.01 turns ON, the display is cleared by SCTRL(048).

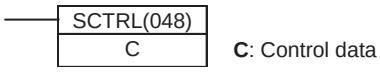


3-25-3 SEVEN-SEGMENT LED CONTROL: SCTRL(048)

Purpose

SCH(047) displays two user-specified digits on the 7-segment display on the CPU Unit according to specifications for individual display segments.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SCTRL(048)
	Executed Once for Upward Differentiation	@SCTRL(048)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control data

Each bit of the control data corresponds to a segment of the two display digits on the CPU Unit.

ON: Light digit
OFF: Do not light digit

Operand Specifications

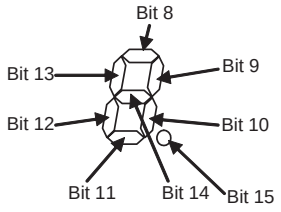
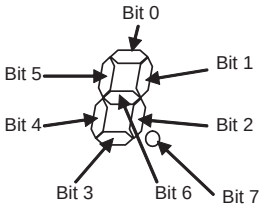
Area	C
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A0 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767

Area	C
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 to *D32767
Constants	#0000 to #FFFF
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

When the execution condition goes ON, SCTRL(048) lights or does not light individual segments of the two digits of the 7-segment display on the CPU Unit according to the value of C. Any combination of segments can be lit.

The display will remain until SCTRL(048) is executed with C set to #0000 to clear all of the display segments. This will also clear displays created with SCH(047). It will not clear displays generated by the system.

Left digit	Right digit
	

Flags

There are no flags affected by SCTRL(048).

Precautions

If displays are created by more than one SCTRL(048) or SCH(047) instruction, the last one that is executed with take priority.

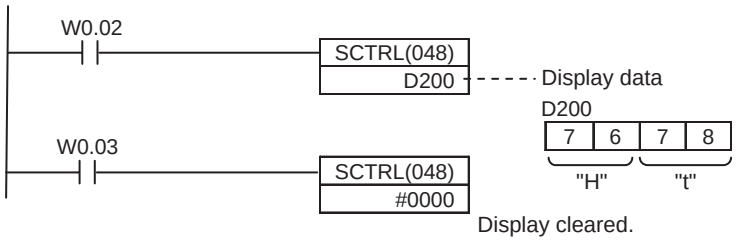
Messages on the 7-segment display on the CPU Unit are displayed in the following order of priority. Therefore, system error message will be given priority even if there is a display created with SCTRL(048) or SCH(047).

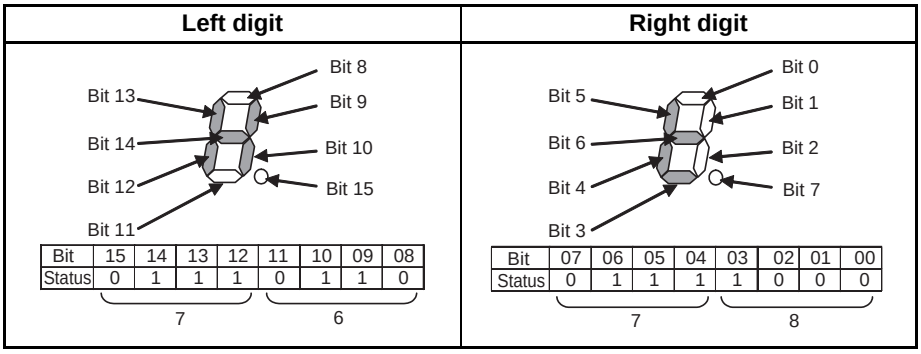
1. Error messages
2. SCTRL(048) or SCH(047) displays
3. Analog adjustment displays
4. Memory Cassette processing displays

If an error occurs while a SCTRL(048) or SCH(047) display is in effect, the error message will be given priority, but the SCTRL(048) or SCH(047) display will return when the error is cleared.

Example

When W0.02 turns ON in the following example, the individual segments of the 7-segment display on the CPU Unit will be controlled according to the contents of D200 (7678 =“Ht”). The display remains even when W0.02 turns OFF. When W0.03 turns ON, the display is cleared.





Numeric Values Corresponding to 7-segment Displays

0 (3F)	1 (06)	2 (5B)	3 (4F)	4 (66)	5 (6D)	6 (7D)	7 (27)	8 (7F)	9 (6F)
A (77)	B (7C)	C (39)	D (5E)	E (79)	F (71)	G (3D)	H (76)	I (19)	J (0D)
K (72)	L (38)	M (55)	N (54)	O (5C)	P (73)	Q (67)	R (50)	S (6D)	T (78)
U (6E)	V (1C)	W (6A)	X (1D)	Y (6E)	Z (49)				

3-26 Clock Instructions

This section describes instructions used with the system clock.

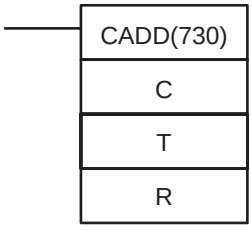
Instruction	Mnemonic	Function code	Page
CALENDAR ADD	CADD	730	916
CALENDAR SUBTRACT	CSUB	731	919
HOURS TO SECONDS	SEC	065	922
SECONDS TO HOURS	HMS	066	925
CLOCK ADJUSTMENT	DATE	735	927

3-26-1 CALENDAR ADD: CADD(730)

Purpose

Adds time to the calendar data in the specified words.

Ladder Symbol



C: First calendar word

T: First time word

R: First result word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	CADD(730)
	Executed Once for Upward Differentiation	@CADD(730)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

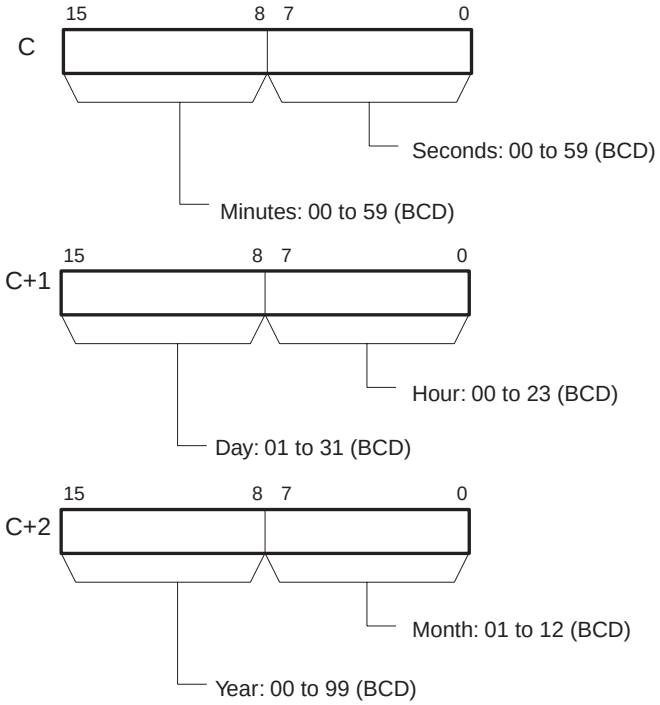
Applicable Program Areas

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

Operands

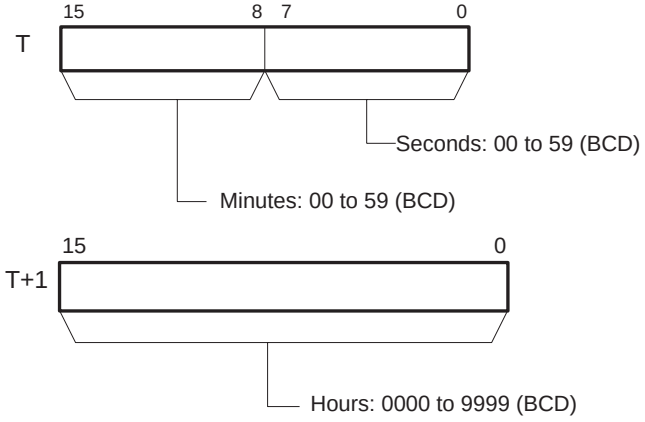
C through C+2: Calendar Data

Set the calendar data in C through C+2 as shown in the following diagram.



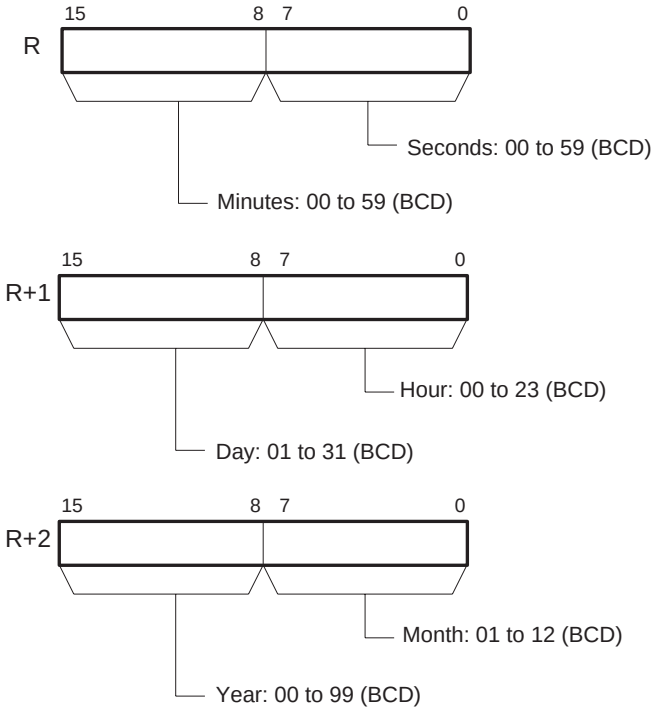
T and T+1: Time Data

Set the time data in T and T+1 as shown in the following diagram.



R through R+2: Result Data

R through R+2 contain the result of the addition.

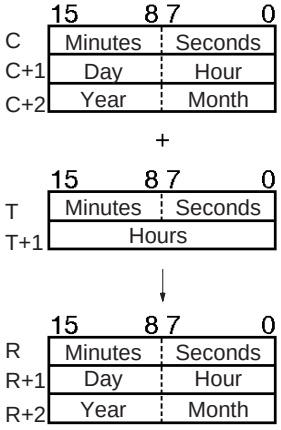


Operand Specifications

Area	C	T	R
CIO Area	CIO 0 to CIO 6141	CIO 0 to CIO 6142	CIO 0 to CIO 6141
Work Area	W0 to W509	W0 to W510	W0 to W509
Holding Bit Area	H0 to H509	H0 to H510	H0 to H509
Auxiliary Bit Area	A0 to A957	A0 to A958	A448 to A957
Timer Area	T0000 to T4093	T0000 to T4094	T0000 to T4093
Counter Area	C0000 to C4093	C0000 to C4094	C0000 to C4093
DM Area	D0 to D32765	D0 to D32766	D0 to D32765
Indirect DM addresses in binary	@D0 to @D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	---
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 IR15 ,IR05+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

CADD(730) adds the calendar data (words C through C+2) to the time data (words T and T+1) and outputs the resulting calendar data to R through R+2.

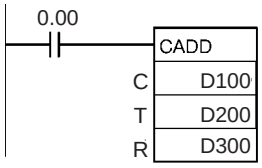


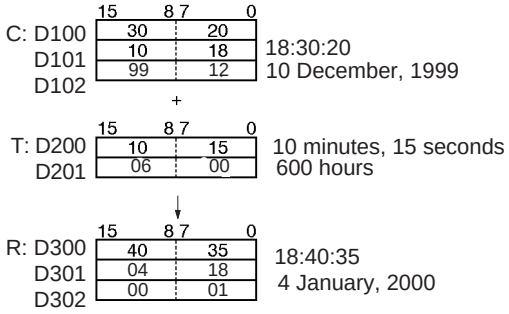
Flags

Name	Label	Operation
Error Flag	ER	ON if the calendar data in C through C+2 is not within the specified ranges. ON if the time data in T and T+1 is not within the specified ranges. OFF in all other cases.

Examples

When CIO 0.00 turns ON in the following example, the calendar data in D100 through D102 (year, month, day, hour, minutes, seconds) is added to the time data in D200 and D201 (hours, minutes, seconds) and the result is output to D300 through D302.



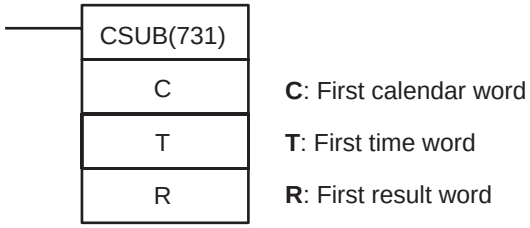


3-26-2 CALENDAR SUBTRACT: CSUB(731)

Purpose

Subtracts time from the calendar data in the specified words.

Ladder Symbol



C: First calendar word

T: First time word

R: First result word

Variations

Variations	**Executed Each Cycle for ON Condition**	CSUB(731)
	Executed Once for Upward Differentiation	@CSUB(731)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

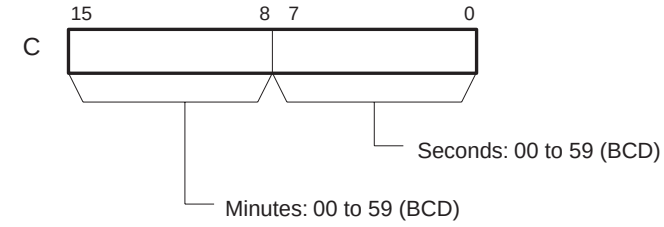
Applicable Program Areas

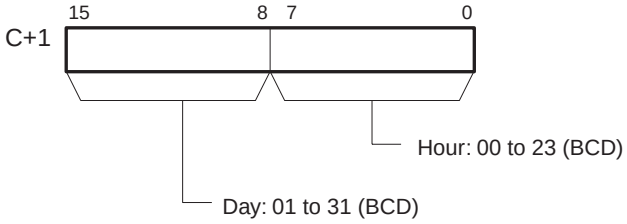
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

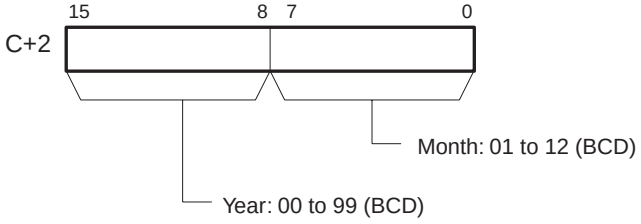
Operands

C through C+2: Calendar Data

Set the calendar data in C through C+2 as shown in the following diagram.

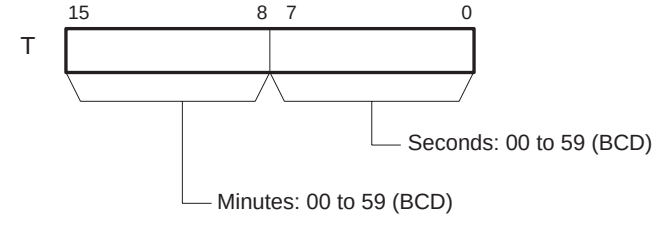


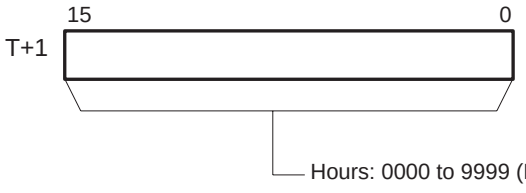




T and T+1: Time Data

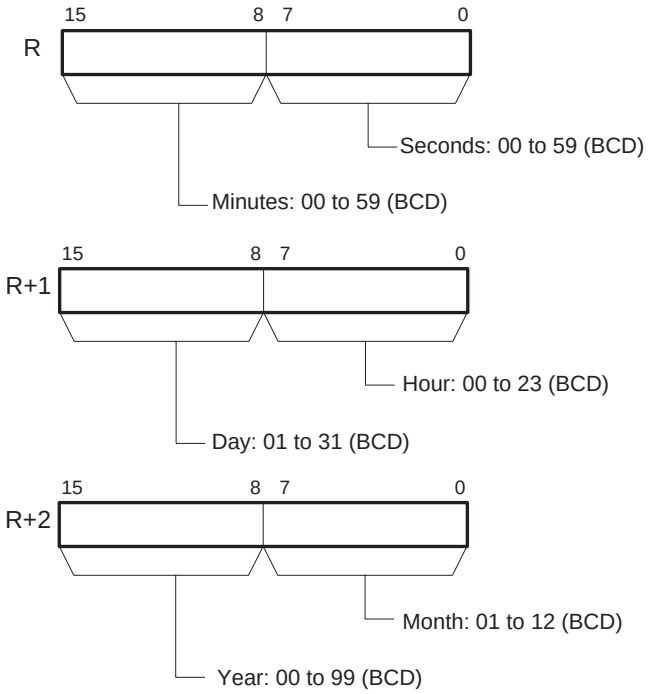
Set the time data in T and T+1 as shown in the following diagram.





R through R+2: Result Data

R through R+2 contain the result of the addition.

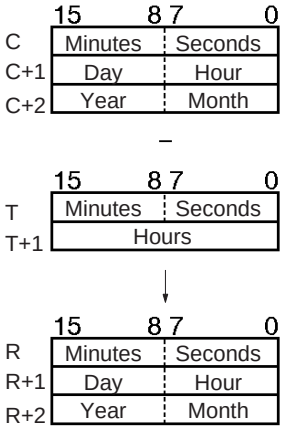


Operand Specifications

Area	C	T	R
CIO Area	CIO 0 to CIO 6141	CIO 0 to CIO 6142	CIO 0 to CIO 6141
Work Area	W0 to W509	W0 to W510	W0 to W509
Holding Bit Area	H0 to H509	H0 to H510	H0 to H509
Auxiliary Bit Area	A0 to A957	A0 to A958	A448 to A957
Timer Area	T0000 to T4093	T0000 to T4094	T0000 to T4093
Counter Area	C0000 to C4093	C0000 to C4094	C0000 to C4093
DM Area	D0 to D32765	D0 to D32766	D0 to D32765
Indirect DM addresses in binary	@D0 to @D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	Specified values only	---
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 IR15 ,IR05+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

CSUB(731) subtracts the time data (words T and T+1) from the calendar data (words C through C+2) to and outputs the resulting calendar data to R through R+2.

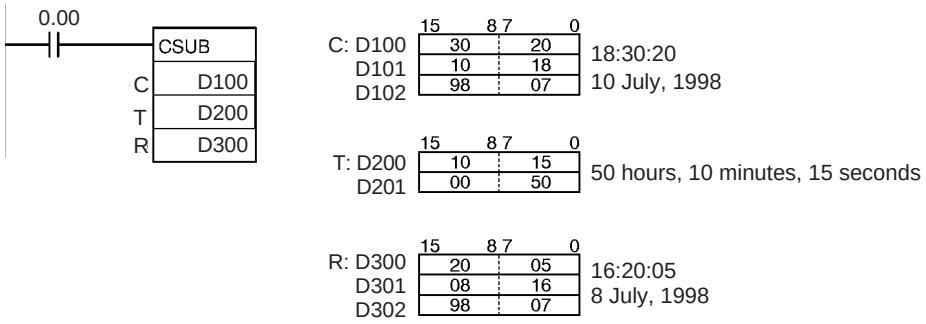


Flags

Name	Label	Operation
Error Flag	ER	ON if the calendar data in C through C+2 is not within the specified ranges. ON if the time data in T and T+1 is not within the specified ranges. OFF in all other cases.

Examples

When CIO 0.00 turns ON in the following example, the time data in D200 and D201 (hours, minutes, seconds) is subtracted from the calendar data in D100 through D102 (year, month, day, hour, minutes, seconds) and the result is output to D300 through D302.

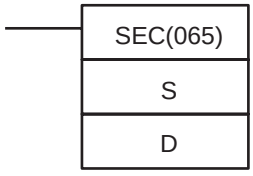


3-26-3 HOURS TO SECONDS: SEC(065)

Purpose

Converts time data in hours/minutes/seconds format to an equivalent time in seconds only.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	SEC(065)
	Executed Once for Upward Differentiation	@SEC(065)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

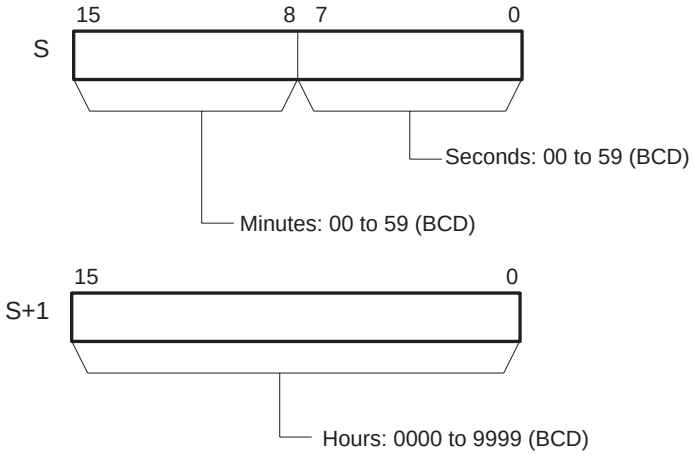
Applicable Program Areas

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

Operands

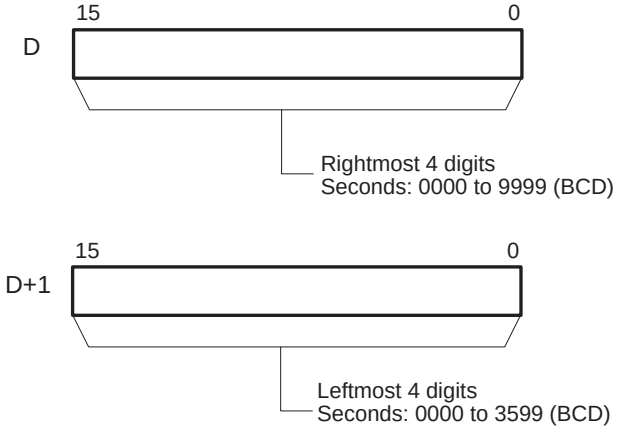
S and S+1: Source Data

Set the hours/minutes/seconds source data in S and S+1, as shown in the following diagram.



D and D+1: Result Data

D and D+1 contain the result data in seconds-only format.



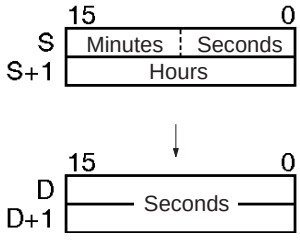
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	

Area	S	D
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	Specified values only	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SEC(065) converts the 8-digit BCD hours/minutes/seconds data in S and S+1 to 8-digit BCD seconds-only data and outputs the result to D and D+1.



Flags

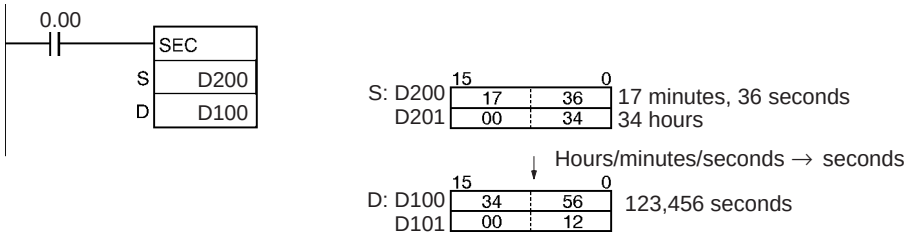
Name	Label	Operation
Error Flag	ER	ON if the minutes data in S (bits 08 to 15) is not BCD and in the range 00 to 59. ON if the seconds data in S (bits 00 to 07) is not BCD and in the range 00 to 59. OFF in all other cases.
Equals Flag	=	ON if the content of D is 0000 after the operation. OFF in all other cases.

Precautions

The maximum value for the source data is 9,999 hours, 59 minutes, and 59 seconds (35,999,999 seconds).

Examples

When CIO 0.00 turns ON in the following example, the hours/minutes/seconds data in D200 and D201 (34 hours, 17 minutes, and 36 seconds) is converted to seconds-only data and the result is output to D100 and D101.

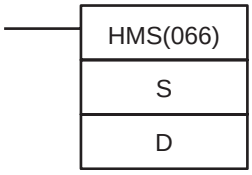


3-26-4 SECONDS TO HOURS: HMS(066)

Purpose

Converts seconds data to an equivalent time in hours/minutes/seconds format.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	HMS(066)
	Executed Once for Upward Differentiation	@HMS(066)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

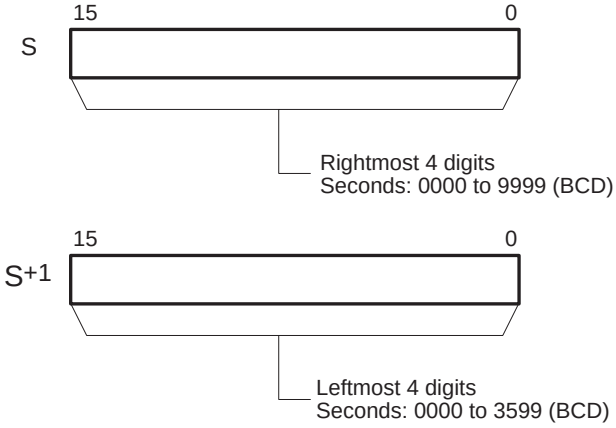
Applicable Program Areas

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

Operands

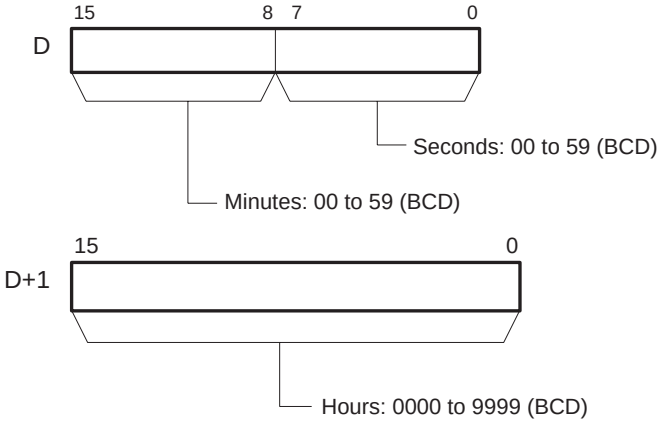
S and S+1: Source Data

Set the seconds source data in S and S+1, as shown in the following diagram.



D and D+1: Result Data

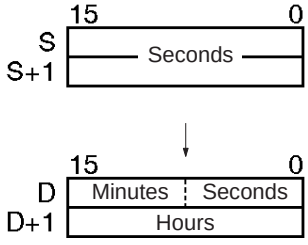
D and D+1 contain the result data in hours/minutes/seconds format.



Area	S	D
CIO Area	CIO 0 to CIO 6142	
Work Area	W0 to W510	
Holding Bit Area	H0 to H510	
Auxiliary Bit Area	A0 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D0 to D32766	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	00000000 to 35999999 (BCD)	---
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

HMS(066) converts the 8-digit BCD seconds-only data in S and S+1 to 8-digit BCD hours/minutes/seconds data and outputs the result to D and D+1.



Flags

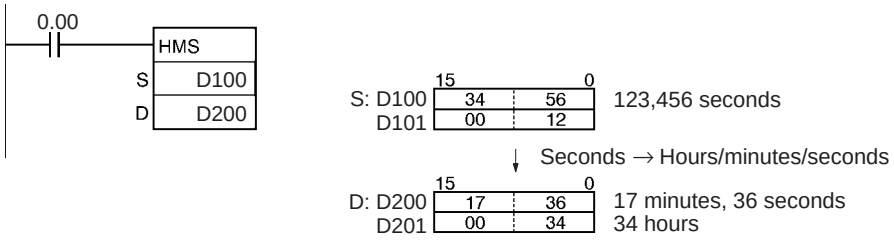
Name	Label	Operation
Error Flag	ER	ON if the seconds data in S and S+1 is not BCD and in the range 0 to 35,999,999. OFF in all other cases.
Equals Flag	=	ON if the content of D is 0000 after the operation. OFF in all other cases.

Precautions

The maximum value for the source data is 35,999,999 seconds (9,999 hours, 59 minutes, and 59 seconds).

Examples

When CIO 0.00 turns ON in the following example, the seconds data in D100 and D101 (123,456 seconds) is converted to hours/minutes/seconds data and the result is output to D200 and D201.



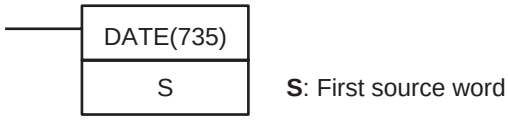
3-26-5 CLOCK ADJUSTMENT: DATE(735)

Purpose

Changes the internal clock setting to the setting in the specified source words.

Note The internal clock setting can also be changed from the CX-Programmer or the CLOCK WRITE FINS command (0702).

Ladder Symbol



S: First source word

Variations

Variations		
	Executed Each Cycle for ON Condition	DATE(735)
	Executed Once for Upward Differentiation	@DATE(735)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

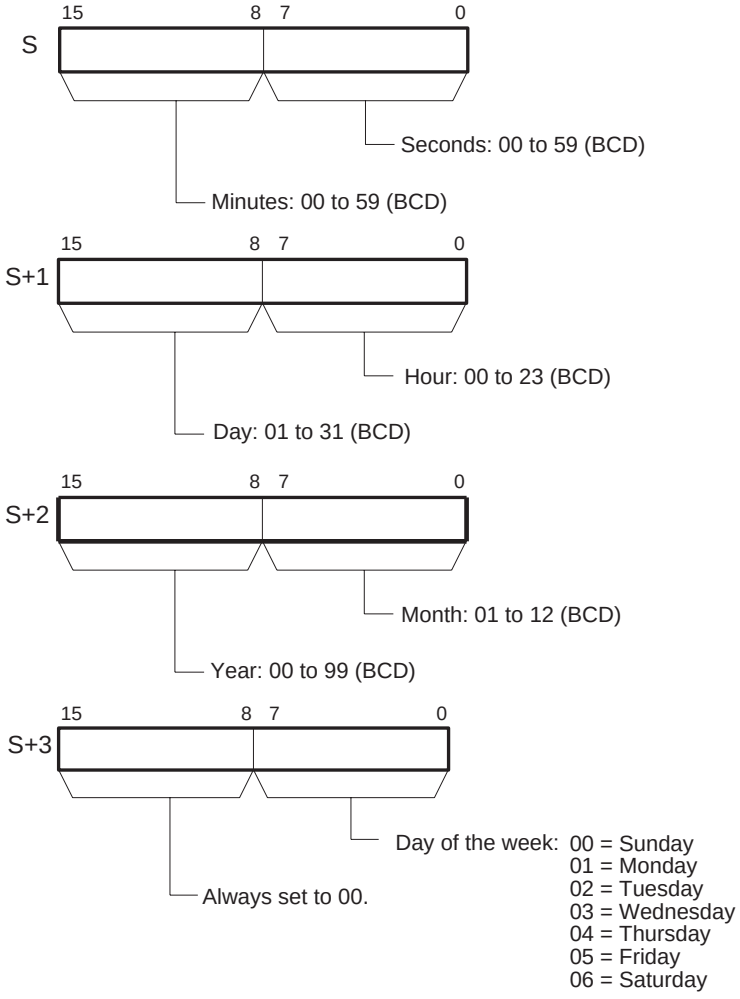
Applicable Program Areas

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

Operands

S through S+3: New Clock Setting

Set the new clock setting in S through S+3 as shown in the following diagram. S through S+3 must be in the same data area.



The following table shows the structure of the Calendar/Clock Area.

Addresses	Contents
A351.00 to A351.07	Second (00 to 59, BCD)
A351.08 to A351.15	Minute (00 to 59, BCD)
A352.00 to A352.07	Hour (00 to 23, BCD)
A352.08 to A352.15	Day of month (01 to 31, BCD)
A353.00 to A353.07	Month (01 to 12, BCD)
A353.08 to A353.15	Year (00 to 99, BCD)
A354.00 to A354.07	Day of week (00 to 06 = Sunday to Saturday, hexadecimal)
A354.08 to A354.15	Always set to 00.

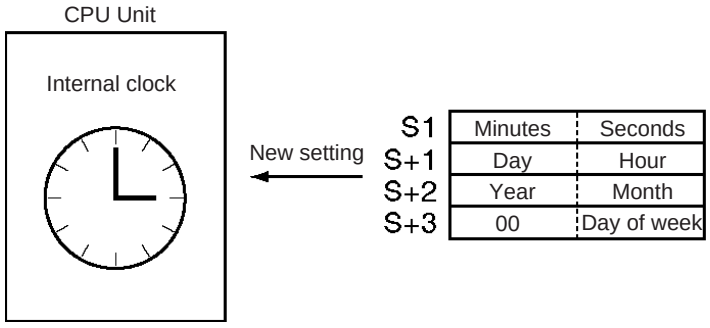
Operand Specifications

Area	S
CIO Area	CIO 0 to CIO 6140
Work Area	W0 to W508]
Holding Bit Area	H0 to H508
Auxiliary Bit Area	A0 to A956
Timer Area	T0000 to T4092

Area	S
Counter Area	C0000 to C4092
DM Area	D0 to D32764
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

DATE(735) changes the internal clock setting according to the clock data in the four source words. The new internal clock setting is immediately reflected in the Calendar/Clock Area (A351 to A354).



Flags

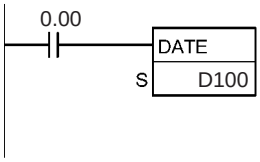
Name	Label	Operation
Error Flag	ER	ON if the new clock setting in S through S+3 is not within the specified range. OFF in all other cases.

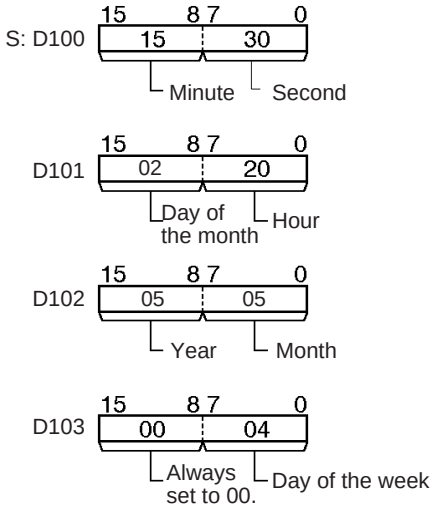
Precautions

An error will not be generated even if the internal clock is set to a non-existent date (such as November 31).

Examples

When CIO 0.00 turns ON in the following example, the internal clock is set to 20:15:30 on Thursday, May 2, 2005.





3-27 Debugging Instructions

This section describes instructions used to debug programs.

Instruction	Mnemonic	Function code	Page
Trace Memory Sampling	TRSM	045	930

3-27-1 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

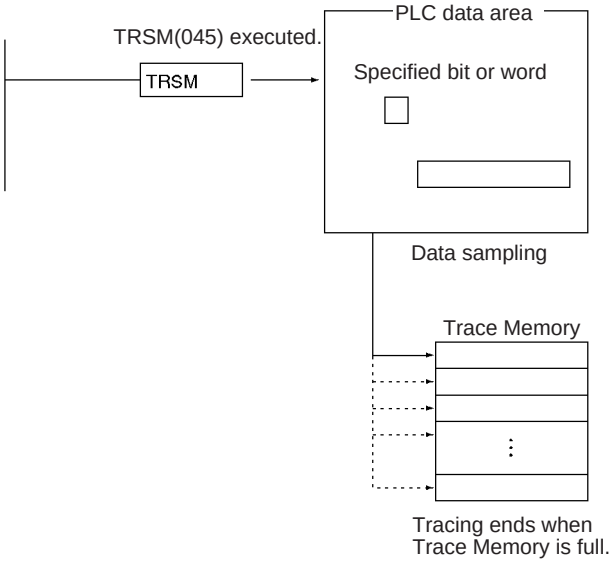
Variations	**Executed Each Cycle**	TRSM(045)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

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.

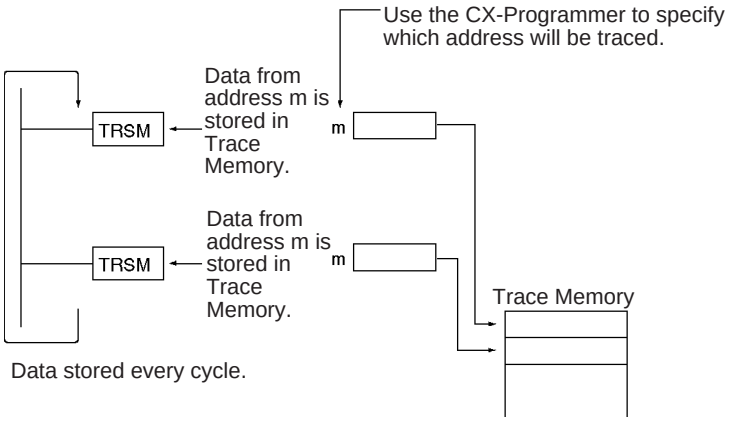


This instruction only indicates when the specified data will be sampled. All other settings and data trace operations are set with the CX-Programmer. The other two ways to control data sampling are sampling at the end of each cycle and sampling at a specified interval (independent of the cycle time).

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's Operation Manual 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 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. Samples will be recorded after the specified delay (positive delay) or the specified number of existing samples will be valid (negative delay).
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 used to determine sampling: 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.

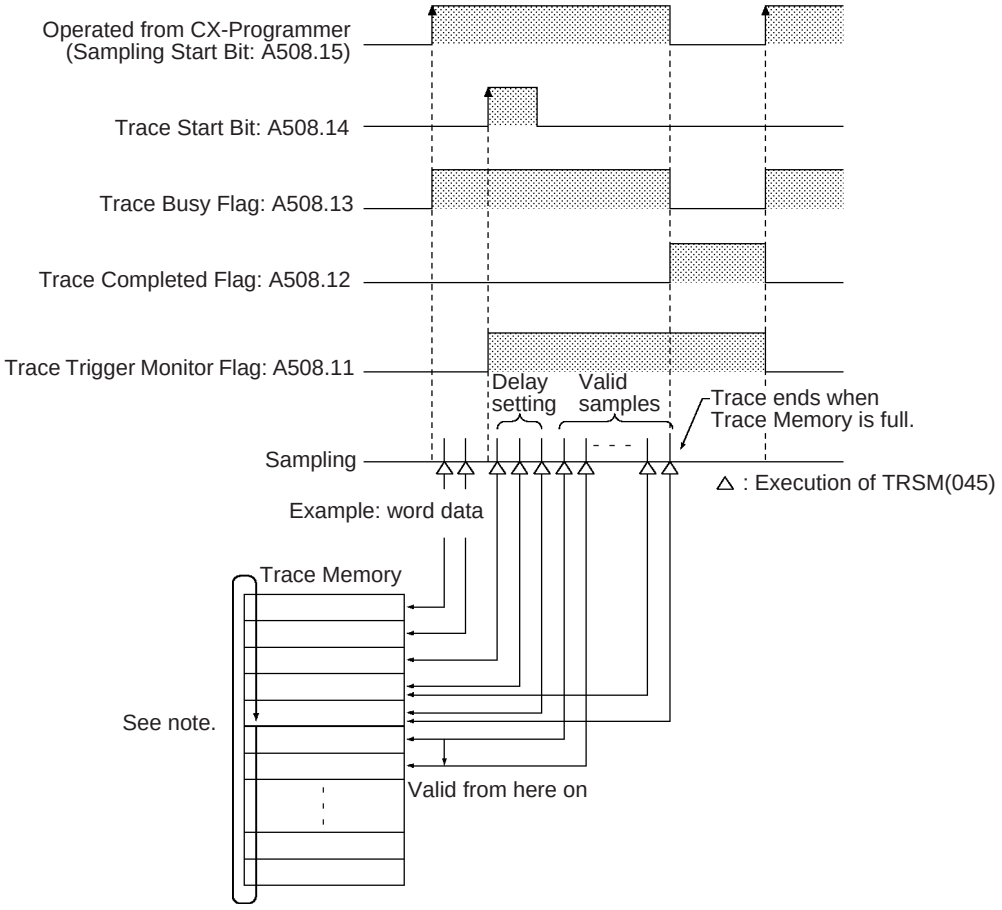
Precautions

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.

3-28 Failure Diagnosis Instructions

This section describes instructions used to define and handle errors.

Instruction	Mnemonic	Function code	Page
FAILURE ALARM	FAL	006	934
SEVERE FAILURE ALARM	FALS	007	942
FAILURE POINT DETECTION	FPD	269	948

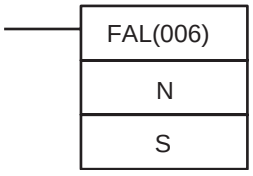
3-28-1 FAILURE ALARM: FAL(006)

Purpose

Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop PLC operation.

Ladder Symbol

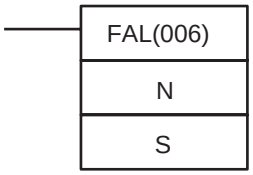
- Generating or Clearing User-defined Non-fatal Errors



N: FAL number

S: First message word or constant (0000 to FFFF)

- Generating Non-fatal System Errors



N: FAL number (value in A529)

S: First word containing the error code and error details

Variations

Variations	**Executed Each Cycle for ON Condition**	FAL(006)
	Executed Once for Upward Differentiation	@FAL(006)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The function of the operands when FAL(006) is used to generate/clear user defined errors is slightly different from the function when FAL(006) is used to generate system errors.

Generating or Clearing User-defined Non-fatal Errors

The following table shows the function of the operands.

Note The value of operand N must be **different from** the content of A529 (the system-generated FAL/FALS number).

N	S	Function
0	#0001 to #01FF	Clears the non-fatal error with the corresponding FAL number.
	#FFFF	Clears all non-fatal errors.
	Other*	Clears the most serious non-fatal error.

N	S	Function
1 to 511 (These FAL numbers are shared with FALS numbers.)	#0000 to #FFFF	Generates a non-fatal error with the corresponding FAL number (no message).
	Word address	Generates a non-fatal error with the corresponding FAL number. The 16-character ASCII message contained in S through S+7 will be displayed on the CX-Programmer.

Note *Other settings would be constants #0200 through #FFFE or a word address.

Generating Non-fatal System Errors

The following table shows the function of the operands.

Note The value of operand N must be **the same as** the content of A529 (the system-generated FAL/FALS number).

Operand	Function
N	1 to 511 (These FAL numbers are shared with FALS numbers.)
S	Error code that will be generated. (See *Description* below.)
S+1	Error details code that will be generated. (See *Description* below.)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	0 to 511	#0000 to #FFFF (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 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. A system error will be generated if the value of N equals the content of A529.

Generating Non-fatal User-defined Errors

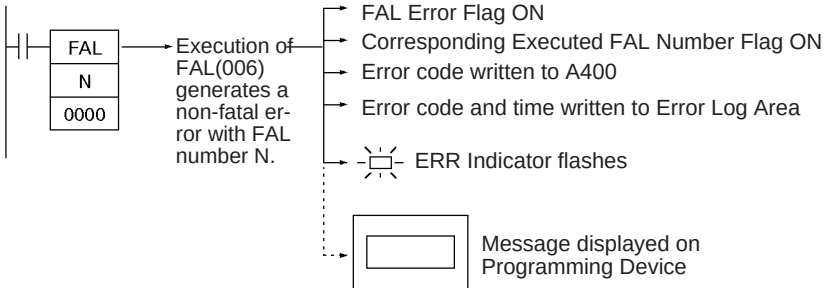
When FAL(006) is executed with N set to an FAL number (&1 to &511) that is not equal to the content of A529 (the system-generated FAL/FALS number), a non-fatal error will be generated with that FAL number and the following processing will be performed:

1,2,3... 1. The FAL Error Flag (A402.15) will be turned ON. (PLC operation will continue.)

2. The Executed FAL Number Flag will be turned ON for the corresponding FAL number. Flags A360.01 to A391.15 correspond to FAL numbers 0001 to 01FF (1 to 511).
3. 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.
4. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).

   Note The error record will not be written to the Error Log Area if the PLC Setup has been set so that errors generated by FAL(006) are not recorded.
5. The ERR Indicator on the CPU Unit will flash.
6. If a word address has been specified in S, the message beginning at S will be registered (displayed on the Programming Device).



The following table shows the error codes and FAL Error Flags for FAL(006).

FAL number	FAL error codes	Executed FAL Number Flags
1 to 511 decimal	4101 to 42FF	A360.01 to A391.15

Displaying Messages with Non-fatal User-defined Errors

If S is a word address and an ASCII message has been stored at S, that message will be displayed at the Programming Device when FAL(006) is executed. (If a message is not required, set S to a constant.)

The message beginning at S will be registered when FAL(006) is executed. Once the message is registered, it will be displayed when the Programming Device is connected.

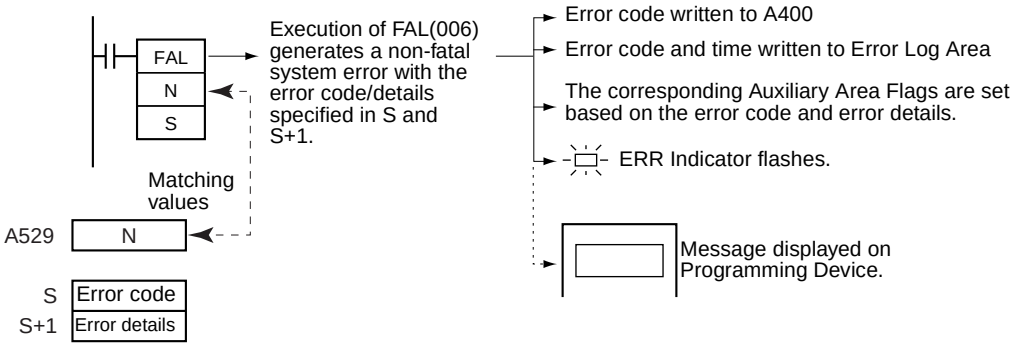
An ASCII message up to 16 characters long can be stored in S through S+7. The leftmost (most significant) byte in each word is displayed first.

The end code for the message is the null character (00 hexadecimal). All 16 characters in words S to S+7 will be displayed if the null character is omitted.

If the contents of the words containing the message are changed after FAL(006) is executed, the message will change accordingly.

Generating Non-fatal System Errors

When FAL(006) is executed with N set to an FAL number (&1 to &511) that is equal to the content of A529 (the system-generated FAL/FALS number), a non-fatal error will be generated with the error code and error details code specified in S and S+1. The following processing will be performed at the same time:



1,2,3...

1. The specified error code will be written to A400.
2. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).
3. The appropriate Auxiliary Area Flags are set based on the error code and error details.
4. The ERR Indicator on the CPU Unit will flash and PLC operation will continue.
5. The non-fatal error message for the specified system error will be displayed on the CX-Programmer.

Note

(1) FAL(006) can be used to generate non-fatal errors from the system when debugging the program. For example, a system error can be generated intentionally to check whether or not error messages are being displayed properly at an interface such as a Programmable Terminal (PT).

(2) The value of A529 (the system-generated FAL/FALS number) is a dummy FAL number (FAL, FALS, and FPD numbers are shared.) used when a non-fatal error is generated intentionally by the system. This number is a dummy FAL number, so it does not change the status of the Executed FAL Number Flags (A360.01 to A391.15) or the error code.
When it is necessary to generate two or more system errors (fatal and/or non-fatal errors), different errors can be generated by executing the FAL/FALS/FPD instructions more than once with the same values in A529 and N, but different values in S and S+1.

(3) If a more serious error (including a system-generated fatal error or FALS(007) error) occurs at the same time as the FAL(006) instruction, the more serious error's error code will be written to A400.

(4) To clear a system error generated by FAL(006), turn the PLC OFF and then ON again. The PLC can be kept ON, but the same processing will be required to clear the error as if the specified error had actually occurred.

The following table shows how to specify error codes and error details in S and S+1.

Error name	S	S+1
Flash Memory Error	00F1 hex	--- (not fixed)
Interrupt Task Error	008B hex	• Bit 15 ON: Interrupt task execution conflicted with Special I/O Unit refreshing Bits 00 to 14: Unit number of Special I/O Unit with refreshing conflict
PLC Setup Error	009B hex	PLC Setup Error Location
Built-in Analog I/O Error	008A hex	--- (not fixed)
CPU Bus Unit Error	0200 hex	CPU Bus Unit's unit number: 0000 to 000F hex
Special I/O Unit Error	0300 hex	Special I/O Unit's unit number:0000 to 005F hex or 00FF hex (unit number undetermined)
Option Board Error	00D0 hex	Option slot number: 0001 or 0002 hex
Battery Error	00F7 hex	--- (not fixed)

Disabling Error Log Entries of User-defined Errors)

Normally when FAL(006) generates a user-defined error, the error code and the time that the error occurred are written to the Error Log Area (A100 through A199). It is possible to set the PLC Setup so that user-defined errors generated by FAL(006) are not recorded in the Error Log.

Even though the error will not be recorded in the Error Log, the FAL Error Flag (A402.15) will be turned ON, the corresponding flag in the Executed FAL Number Flags (A360.01 to A391.15) will be turned ON, and the error code will be written to A400.

Disable Error Log entries for user-defined FAL(006) errors when you want to record only the system-generated errors. For example, this function is useful during debugging if the FAL(006) instructions are used in several applications and the Error Log is becoming full of user-defined FAL(006) errors. The following table shows the PLC Setup setting:

Item	Setting
Name	FAL Error Log Registration
Settings	0: Record FAL Errors in Error Log. 1: Do not record FAL Errors in Error Log.
Default setting	0: Record FAL Errors in Error Log.
Times that PLC Setup setting is read	Every cycle (when an FAL Error occurs)

Even if PLC Setup word 129 bit 15 is set to 1 (Do not record FAL Errors in Error Log.), the following errors will be recorded:

- Fatal errors generated by FALS(007)
- Non-fatal errors from the system
- Fatal errors from the system
- Non-fatal errors from the system generated intentionally with FAL(006) or FPD(269)
- Fatal errors from the system generated intentionally with FALS(007)

Clearing Non-fatal Errors without the CX-Programmer

1. Clearing User-defined Non-fatal Errors
   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. When more than one FAL error has occurred, the FAL error with the smallest FAL number will be cleared.

2. Clearing Non-fatal System Errors
   There are two ways to clear non-fatal system errors generated with FAL(006).
   - Turn the PLC OFF and then ON again.
   - When keeping the PLC ON, the system error must be cleared as if the specified error had actually occurred.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 511 decimal. ON if a non-fatal system error is being generated, but the specified error code or error details code is incorrect. 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
FAL Error Flag	A402.15	ON when an error is generated with FAL(006).
Executed FAL Number Flags	A360.01 to A391.15	When an error is generated with FAL(006), the corresponding flag will be turned ON. Flags A360.01 to A391.15 correspond to FAL numbers 0001 to 01FF.

- Auxiliary Area Words/Flags for System Errors Only

Name	Address	Operation
System-generated FAL/FALS number	A529	A dummy FAL/FALS number is used when a system error is generated with FAL(006). Set the same dummy FAL/FALS number in this word (0001 to 01FF hex, 1 to 511 decimal).

- 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 and time/date of occurrence 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 between 0000 and 01FF. An error will occur and the Error Flag will be turned ON if N is outside of the specified range.

Examples

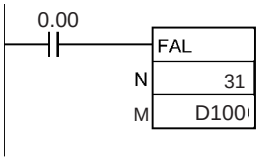
Generating a Non-fatal Error

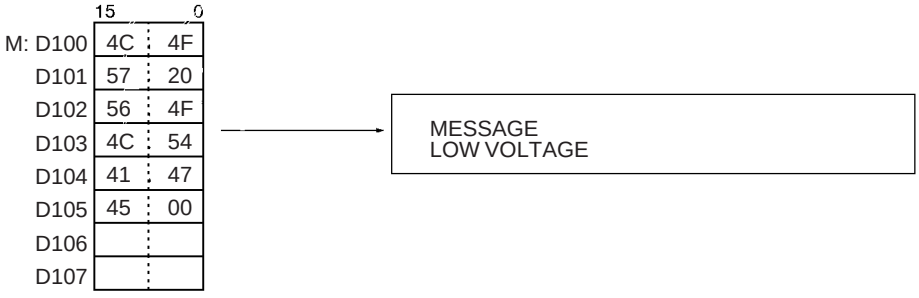
When CIO 0.00 is ON in the following example, FAL(006) will generate a non-fatal error with FAL number 31 and execute the following processes.

1,2,3...

1. The FAL Error Flag (A402.15) will be turned ON.
2. The corresponding Executed FAL Number Flag (A361.14) will be turned ON.
3. The corresponding error code (411F) will be written to A400.

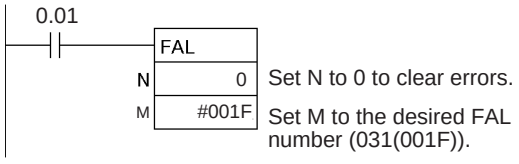
   Note If two or more errors occur at the same time, the error code of the most serious error (with the highest error code) will be stored in A400.
4. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
5. The ERR Indicator on the CPU Unit will flash.
6. The ASCII message in D100 to D107 will be displayed at the Programming Device. (If a message is not required, specify a constant for S.)





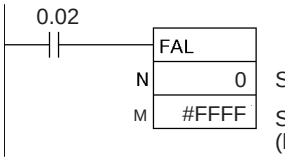
Clearing a Particular Non-fatal Error

When CIO 0.01 is ON in the following example, FAL(006) will clear the non-fatal error with FAL number 31, turn OFF the corresponding Executed FAL Number Flag (A361.14), and turn OFF the FAL Error Flag (A402.15).



Clearing All Non-fatal Errors

When CIO 0.02 is ON in the following example, FAL(006) will clear all of the non-fatal errors, turn OFF the Executed FAL Number Flags (A360.01 to A391.15), and turn OFF the FAL Error Flag (A402.15).

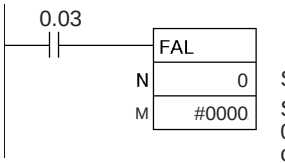


Set N to 0 to clear errors.

Set M to FFFF to clear all non-fatal errors (both FAL(006) and system errors).

Clearing the Most Serious Non-fatal Error

When CIO 0.03 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 corresponding Executed FAL Number Flag and the FAL Error Flag (A402.15) will be turned OFF.



Set N to 0 to clear errors.

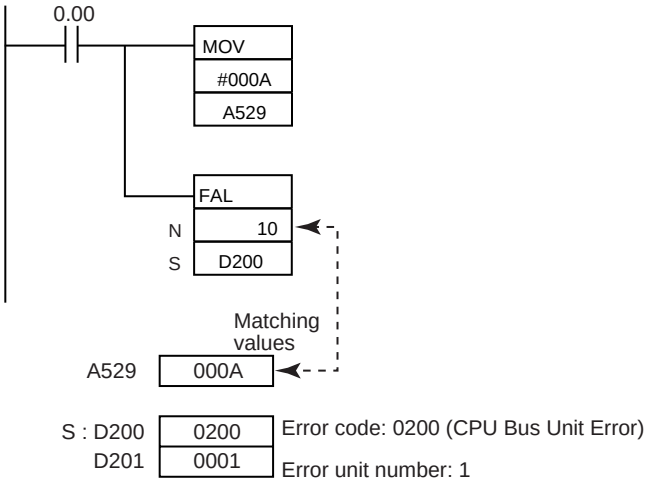
Set M to 0000, another constant between 0200 and FFFE, or a word address to clear the most serious non-fatal error. (In this case, M is set to 0000.)

Generating a Non-fatal System Error

When CIO 0.00 is ON in the following example, FAL(006) will generate a CPU Bus Unit Setup Error for unit number 1. In this case, dummy FAL number 10 is used and the corresponding value (000A hex) is stored in A529.

1,2,3...

1. The specified error code (0200) will be written to A400 if it is the most serious error.
2. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
3. The CPU Bus Unit Error Flag (A402.07) and CPU Bus Unit Error Flag for unit number 1 (A417.01) will be turned ON.
4. The CPU Unit's ERR Indicator will flash.
5. A message (CPU BU ST ERR 01) will be displayed at the Programming Device indicating that an error has occurred with CPU Bus Unit 1.



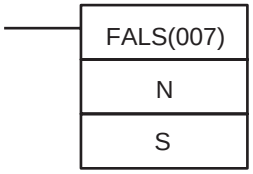
3-28-2 SEVERE FAILURE ALARM: FALS(007)

Purpose

Generates user-defined fatal errors. Fatal errors stop PLC operation.

Ladder Symbol

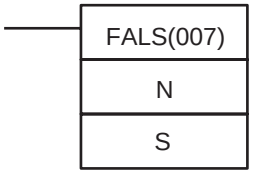
- Generating User-defined Fatal Errors



N: FALS number

S: First message word or constant (0000 to FFFF)

- Generating Fatal System Errors



N: FALS number (value in A529)

S: First word containing the error code and error details

Variations

Variations	**Executed Each Cycle for ON Condition**	FALS(007)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

Generating User-defined Fatal Errors

The following table shows the function of the operands.

Note The value of operand N must be **different from** the content of A529 (the system-generated FAL/FALS number).

Operand	Function
N	1 to 511 (These FALS numbers are shared with FAL numbers.)
S	Specifies the first of eight words containing an ASCII message to be displayed on the CX-Programmer. Specify a constant (0000 to FFFF) if a message is not required.

Generating Fatal Errors from the System

The following table shows the function of the operands.

Note The value of operand N must be **the same as** the content of A529 (the system-generated FAL/FALS number).

Operand	Function
N	1 to 511 (These FALS numbers are shared with FAL numbers.)
S	Error code that will be generated. (See *Description* below.)
S+1	Error details code that will be generated. (See *Description* below.)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511

Area	N	S
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	Specified values only	#0000 to #FFFF (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 IR15 ,IR+(++)0 to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

FALS(007) generates a fatal error. FALS(007) can also be used to generate fatal system errors as well as fatal user-defined errors. (A system error will be generated if the value of N equals the content of A529.)

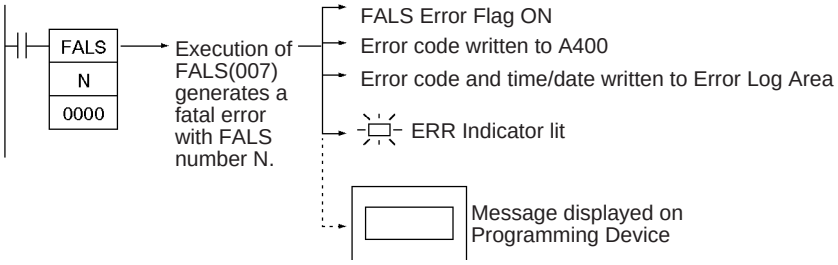
Generating Fatal User-defined Errors

When FALS(007) is executed with N set to an FALS number (1 to 511) that is not equal to the content of A529 (the system-generated FAL/FALS number), a fatal error will be generated with that FALS number and the following processing will be performed:

1,2,3...

1. The FALS Error Flag (A401.06) will be turned ON. (PLC 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 and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
4. The ERR Indicator on the CPU Unit will be lit.
5. If a word address has been specified in S, the ASCII message beginning at S will be registered (displayed on the Programming Device).



The following table shows the error codes for FALS(007).

FALS number	FALS error codes
1 to 511	C101 TO C2FF

Displaying Messages with Fatal User-defined Errors

If S is a word address, the ASCII message beginning at S will be displayed at the Programming Device when FALS(007) is executed. (If a message is not required, set S to a constant.)

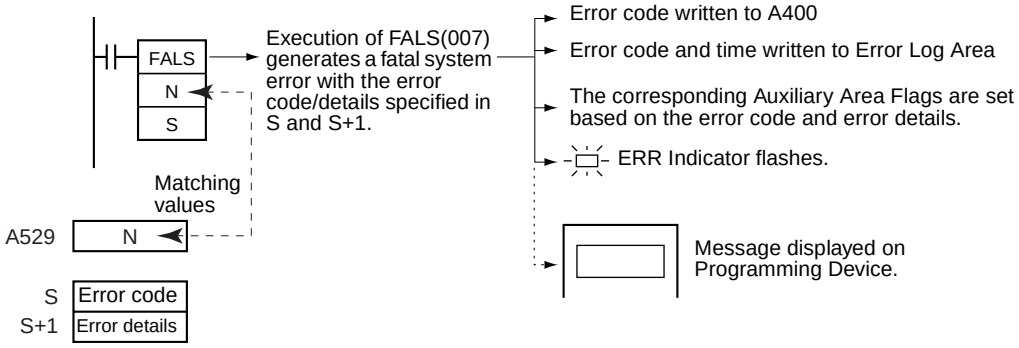
The message beginning at S will be registered when FALS(007) is executed. Once the message is registered, it will be displayed when the Programming Device is connected.

An ASCII message up to 16 characters long can be stored in S through S+7. The leftmost (most significant) byte in each word is displayed first.

The end code for the message is the null character (00 hexadecimal). All 16 characters in words S to S+7 will be displayed if the null character is omitted.

If the contents of the words containing the message are changed after FALS(007) is executed, the message will change accordingly.

Generating Non-fatal System Errors



When FALS(007) is executed with N set to an FAL number (1 to 511) that is equal to the content of A529 (the system-generated FAL/FALS number), a fatal error will be generated with the error code and error details code specified in S and S+1. The following processing will be performed at the same time:

1,2,3...

1. The specified error code will be written to A400.
2. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).
3. The appropriate Auxiliary Area Flags are set based on the error code and error details.
4. The ERR Indicator on the CPU Unit will light and PLC operation will be stopped.
5. The fatal error message for the specified system error will be displayed on the Programming Device.

Note (1) The value of A529 (the system-generated FAL/FALS number) is a dummy FAL number (FAL, FALS, and FPD numbers are shared.) used when a non-fatal error is generated intentionally by the system. This number is a dummy FAL number, so it is not reflected in the error code.
When it is necessary to generate two or more system errors, different errors can be generated by executing the FAL/FALS/FPD instructions more

than once with the same values in A529 and N, but different values in S and S+1.

(2) If a more serious error (including a system-generated fatal error or another FALS(007) error) occurs at the same time as the FALS(007) instruction, the more serious error's error code will be written to A400.

(3) To clear a system error generated by FALS(007), turn the PLC OFF and then ON again. The PLC can be kept ON, but the same processing will be required to clear the error as if the specified error had actually occurred. Refer to information on troubleshooting in the *CP Series Operation Manual* for details.

(4) The following table shows how the IOM Hold Bit affects the status of I/O memory and the status of outputs on Output Units after a fatal system error has been generated with FALS(007).

IOM Hold Bit (A500.12)	Status of I/O memory	Status of outputs on Output Units
ON	Retained	OFF
OFF	Cleared	OFF

Note Unlike user-defined fatal errors, system errors generated by FALS(007) will clear I/O memory if the IOM Hold Bit is OFF. The following areas will be cleared: CIO Area, Work Area, Timer Flags and PVs, Index Registers, and Data registers.

The following table shows how to specify error codes and error details in S and S+1.

Error name	S	S+1
	Error code	Error details
Memory Error	80F1 hex	• Bits 00 to 09: Memory Error Location Bit 00: User program Bit 04: PLC Setup Bit 07: Routing table Bit 08: CPU Bus Unit Setup Bit 09: Memory Cassette transfer error • Bits 10 to 15: Invalid
I/O Bus Error	80C0 hex	0A0A hex: CPM1A Expansion Unit/Expansion I/O Unit error 0000 hex: CJ-series Unit error (first Unit) 0001 hex: CJ-series Unit error (second Unit) 0F0F hex: CJ-series Unit error (unknown Unit) 0E0E hex: CJ-series Unit error (no End Cover)
Unit Number Duplication Error	80E9 hex	CPU Bus Unit's duplicated unit number 0000 to 000F hex
		Special I/O Unit's duplicated unit number 8000 to 805F hex

Error name	S	S+1
	Error code	Error details
Too Many I/O Points Error	80E1 hex	Bits 13 to 15: Error Cause Bits 00 to 12: Details • Too many words for CPM1A Expansion Units/ Expansion I/O Units Bits 13 to 15: 010 Bits 00 to 12: All zeroes • Too many CPM1A Expansion Units/Expansion I/O Units Bits 13 to 15: 011 Bits 00 to 12: All zeroes • Too many words for CJ-series Units Bits 13 to 15: 111 Bits 00 to 12: All zeroes
I/O Table Setting Error	80E0 hex	--- (Not fixed.)
Program Error	80F0 hex	• Bits 08 to 15: Error Cause Bit 15: UM overflow error Bit 14: Illegal instruction error Bit 13: Differentiation overflow error Bit 12: Task error Bit 11: No END error Bit 10: Illegal access error Bit 09: Indirect DM BCD error Bit 08: Instruction error • Bits 00 to 07: Invalid
Cycle Time Overrun Error	809F hex	--- (Not fixed.)

Clearing FALS(007) Fatal System Errors

There are two ways to clear fatal system errors generated with FALS(007).

1. Turn the PLC OFF and then ON again.
2. When keeping the PLC ON, the system error must be cleared as if the specified error had actually occurred.

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 Programming Device or turn the PLC OFF and then ON again.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0001 to 01FF (1 to 511 decimal). ON if a fatal system error is being generated, but the specified error code or error details code is incorrect. 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 System Errors Only

Name	Address	Operation
System-generated FAL/FALS number	A529	A dummy FAL/FALS number is used when a system error is generated with FALS(007). Set the same dummy FAL/FALS number in this word (0001 to 01FF hex, 1 to 511 decimal).

- 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 and time/date of occurrence 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

The end code for the message is the null character (00 hexadecimal). All 16 characters in words S to S+7 will be displayed if the null character is omitted.

N must 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

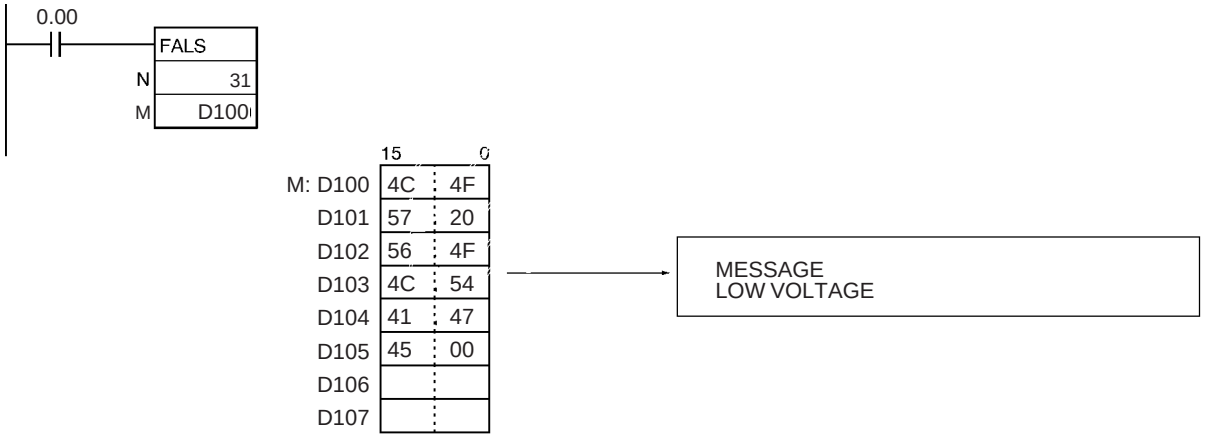
Generating a User-defined Error

When CIO 0.00 is ON in the following example, FALS(007) will generate a fatal error with FAL 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 and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
4. The ERR Indicator on the CPU Unit will be lit.
5. The ASCII message in D100 to D107 will be displayed at the Programming Device. (If a message is not required, specify a constant for S.)

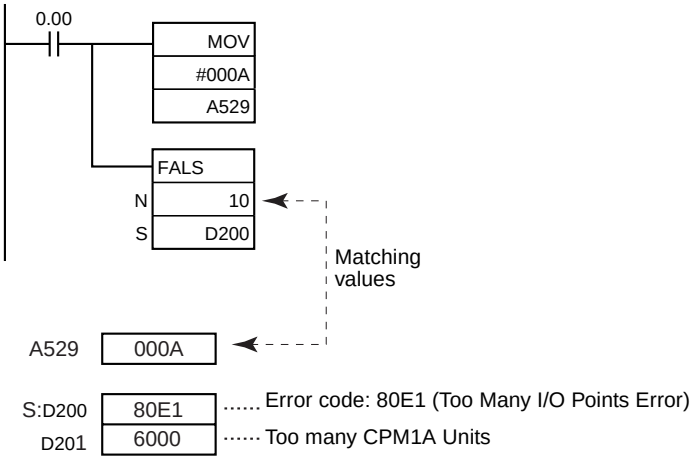


Generating a Non-fatal System Error

When CIO 0.00 is ON in the following example, FALS(007) will generate a Too Many I/O Points Error (too many CPM1A Expansion Units/Expansion I/O Units). In this case, dummy FAL number 10 is used and the corresponding value (000A hex) is stored in A529.

1,2,3...

1. The specified error code (80E1) will be written to A400 if it is the most serious error.
2. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
3. The Too Many I/O Points Flag (A401.11) will be turned ON.
4. The CPU Unit's ERR Indicator will light and PLC operation will stop.
5. A message (TOO MANY I/O PNT) will be displayed at the CX-Programmer indicating that a Too Many I/O Points Error has occurred.

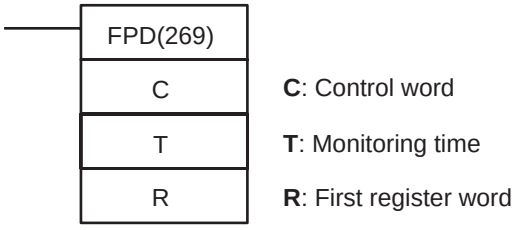


3-28-3 FAILURE POINT DETECTION: FPD(269)

Purpose

Diagnoses a failure in an instruction block by monitoring the time between execution of FPD(269) and execution of a diagnostic output and finding which input is preventing an output from being turned ON.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	FPD(269)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

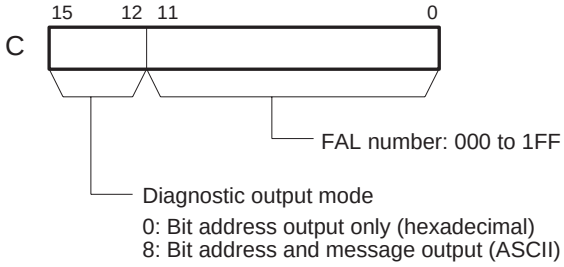
Applicable Program Areas

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

Operands

C: Control Word

C must be a constant between 0000 and 01FF or between 8000 and 81FF. The following diagram shows the function of the digits in the control word.



T: Monitoring Time

T must be between 0 and 9,999 decimal (between 0000 and 270F hex). A value of 0 disables time monitoring; values in the range of 1 to 270F set the monitoring time from 0.1 to 999.9 seconds.

R: First Register Word

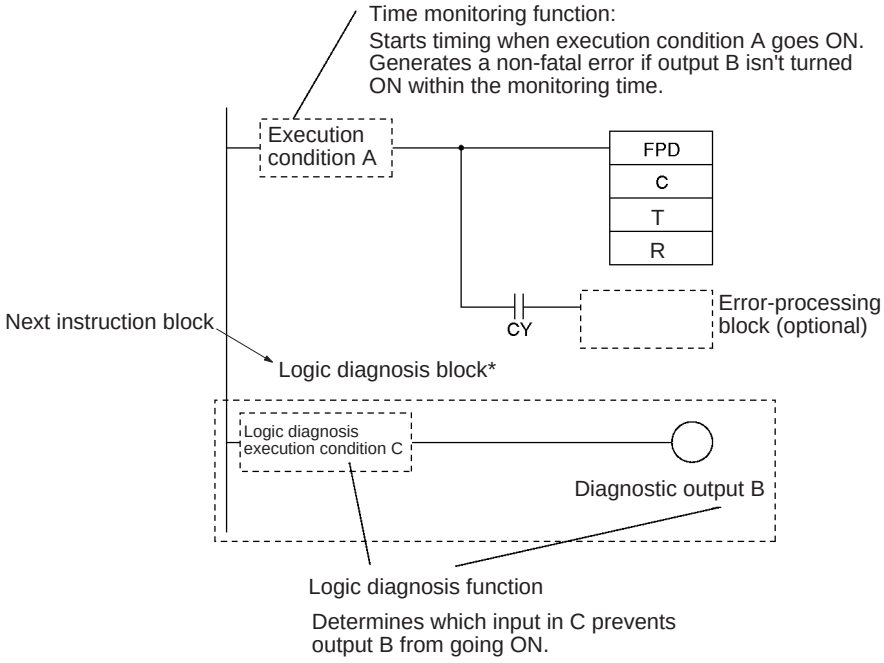
The functions of the register words are described on page 952.

Operand Specifications

Area	C	T	R
CIO Area	---	CIO 0 to CIO 6143	
Work Area	---	W0 to W511	
Holding Bit Area	---	H0 to H511	
Auxiliary Bit Area	---	A0 to A447 A448 to A959	A448 to A959
Timer Area	---	T0000 to T4095	
Counter Area	---	C0000 to C4095	
DM Area	---	D0 to D32767	
Indirect DM addresses in binary	---	@ D0 to @ D32767	
Indirect DM addresses in BCD	---	*D0 to *D32767	
Constants	Specified values only	#0000 to #270F (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 IR15	

Description

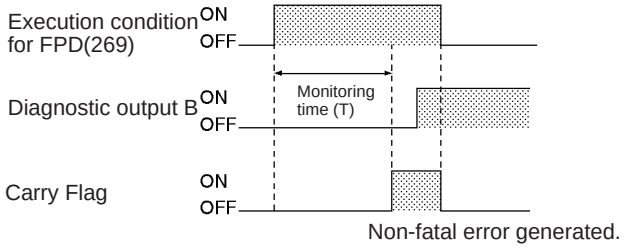
FPD(269) performs time monitoring and logic diagnosis. The time monitoring function generates a non-fatal error with the specified FAL number if the diagnostic output is not turned ON within the specified monitoring time. The logic diagnosis function indicates which input is preventing the output from being turned ON.



Note *The logic diagnosis block begins with the first LD (not LD TR) or LD NOT instruction after FPD(269) and ends with the first OUT (not OUT TR) or other right-hand instruction.

Time Monitoring Function

FPD(269) starts timing when it is executed (when execution condition A goes ON); it will generate a non-fatal error and turn ON the Carry Flag if the diagnostic output is not turned ON within the specified monitoring time.



Note The diagnostic output must go ON within the monitoring time. The teaching function can be used set the monitoring time automatically.

The following processing will be performed when the Carry Flag is turned ON. (This processing will not be performed if the FAL number is set to 000 in C.)

1,2,3...

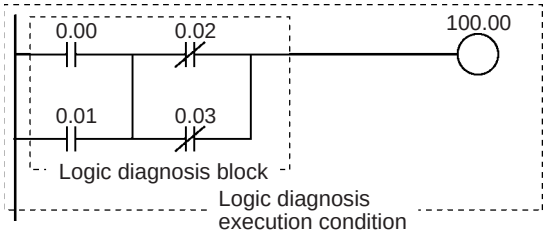
1. The FAL Error Flag (A402.15) will be turned ON. (PLC operation continues.)
2. The Executed FAL Number Flag for the specified FAL number will be turned ON. (Flags A360.01 to A391.15 correspond to FAL numbers 001 to 1FF.)

3. The corresponding error code will be written to A400. Error codes 4101 to 42FF correspond to FAL numbers 001 to 1FF.
   (If a more serious error has occurred (one with a higher error code) at the same time, the error code of the more serious error will be stored in A400.)
4. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
5. The ERR Indicator on the CPU Unit will flash.
6. If the output mode has been set for bit address and message output (leftmost digit of C set to 8), the ASCII message stored in R+2 through R+10 will be displayed as a non-fatal error message.

Logic Diagnosis Function

Every cycle that the execution condition for FPD(269) is ON, FPD(269) determines which input bit is causing the diagnostic output to be OFF and writes the bit's address to the register area beginning at R.

If input bits CIO 0.00 through CIO 0.03 are all ON in the following example, FPD(269) would determine that the normally closed CIO 0.02 condition is causing output CIO 100.00 to remain OFF. FPD(269) would turn ON the Bit Address Found Flag (bit 15 of R) and write the bit address to register words R+2 to R+4.



The logic diagnosis function is executed every cycle as long as the execution condition for FPD(269) is ON. The operation of the logic diagnosis function is independent of the time monitoring function.

When two or more input bits are preventing the diagnostic output from being turned ON, the address of the first input bit in the execution condition (on the highest instruction line and nearest the left bus bar) will be output to R+2 through R+4.

Input bits in LD, LD NOT, AND, AND NOT, OR, and OR NOT instructions (including differentiated and immediate-refreshing variations) will be checked by the logic diagnosis function. Input bits in other instructions and operands addressed indirectly through Index Registers will not be checked.

The logic diagnosis block begins with the first LD (not LD TR) or LD NOT instruction after FPD(269) and ends with the first OUT (not OUT TR) or other right-hand instruction.

There are two diagnostic output modes, set with the leftmost digit of C.

1,2,3... 1. Bit address output mode (Leftmost digit of C = 0)

   Bit 15 of R (the Bit Address Found Flag) is turned ON when an input bit address has been found and bit 14 of R indicates whether the input is normally ON or normally OFF.
   The 8-digit hexadecimal PLC memory address of the input bit is output to R+3 and R+2.
2. Bit address and message output mode (Leftmost digit of C = 8)

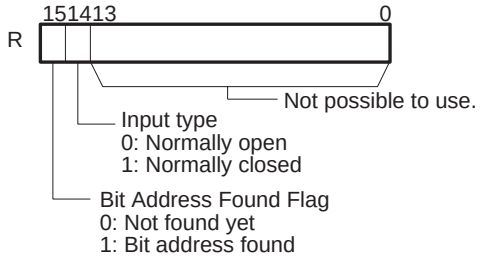
Bit 15 of R (the Bit Address Found Flag) is turned ON when an input bit address has been found and bit 14 of R indicates whether the input is normally ON or normally OFF.
The input bit's address is output to R+2 through R+4 as 6 ASCII characters.

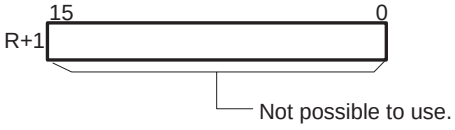
Register Word Functions

The register words contain the results of the diagnostic function and can also contain an ASCII error message which is displayed when an error is generated by the time monitoring function. The function of the register words depends upon the diagnostic output mode which is set with the leftmost digit of C.

Bit Address Output (C=0□□□)

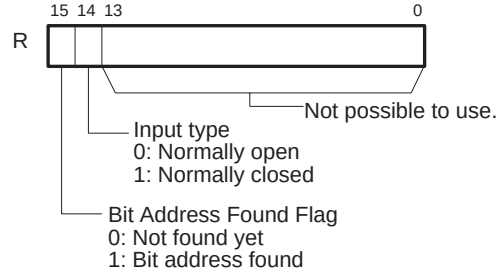
When the leftmost digit of C is set to 0, the 8-digit hexadecimal PLC memory address of the input bit is output to R+2 and R+3. R contains two flags which indicate whether an input bit has been found and whether it is used in a normally open or normally closed input condition.

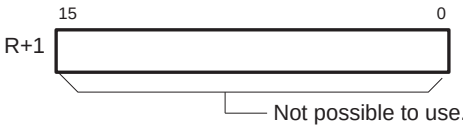




Bit Address and Message Output (C=8□□□)

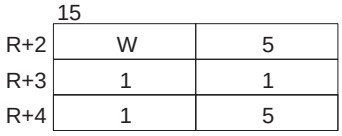
When the leftmost digit of C is set to 8, the ASCII address of the input bit is output to R+2 to R+4. R contains two flags which indicate whether an input bit has been found and whether it is used in a normally open or normally closed input condition.





Register words R+2 to R+4 indicate the address of the input which prevented the diagnostic output from being turned ON. The bit address is output to these words in ASCII. The following table shows the ASCII representations for each area.

Area	ASCII text	Notes
Auxiliary Area	A0.00 to A959.15	---
Holding Area	H0.00 to H511.15	---
Work Area	W0.00 to W511.15	---
CIO Area	0.00 to 6143.15	---
Task Flags	TK0 to TK1023	---
Timer Area	_T0 to _T4095	The "_" represents an ASCII space. (Character code 20.)
Counter Area	_C0 to _C4095	

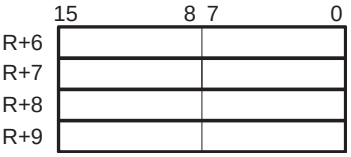


Bit address written in ASCII

Register words R+2 through R+5 would have the following values for W511.15:

Word	Bits 8 to 15	Bits 0 to 7
R+2	W	5
R+3	1	1
R+4	1	5
R+5	2D (hexadecimal)	Input type (hexadecimal) 30: Normally open 31: Normally closed

The user can store an ASCII message in register words R+6 to R+9. This message will be displayed on the CX-Programmer if a non-fatal error is generated by the time monitoring function. Mark the end of the message with the null character (00 hexadecimal).



Disabling Error Log Entries of Non-fatal FPD(269) Errors

Normally when the FPD(269) Time Monitoring Function generates a non-fatal error, the error code and the time that the error occurred are written to the Error Log Area (A100 through A199). It is possible to set the PLC Setup so that the non-fatal errors generated by FPD(269) are not recorded in the Error Log.

Even though the error will not be recorded in the Error Log, the FAL Error Flag (A402.15) will be turned ON, the corresponding flag in the Executed FAL Number Flags (A360.01 to A391.15) will be turned ON, and the error code will be written to A400.

Disable Error Log entries for FPD(269) time-monitoring errors when you want to record only the system-generated errors. For example, this function is useful during debugging if the FPD(269) and FAL(006) instructions are used in several applications and the Error Log is becoming full of these errors. The following table shows the PLC Setup setting:

Item	Setting
Name	FAL Error Log Registration
Settings	0: Record FAL Errors in Error Log. 1: Do not record FAL Errors in Error Log.
Default setting	0: Record FAL Errors in Error Log.
Times that PLC Setup setting is read	Every cycle (when an FAL Error occurs)

Even if PLC Setup word 129 bit 15 is set to 1 (Do not record FAL Errors in Error Log.), the following errors will be recorded:

- Fatal errors generated by FALS(007)
- Non-fatal errors from the system
- Fatal errors from the system
- Non-fatal errors from the system generated intentionally with FAL(006)
- Fatal errors from the system generated intentionally with FALS(007)

Setting Monitoring Time with the Teaching Function

If a word address is specified for T, the monitoring time can be set automatically with the teaching function. Use the following procedure when a word address has been set for T.

1,2,3...

1. Turn ON the FPD Teaching Bit (A598.00).
2. FPD(269) will measure the time from the point when the execution condition for FPD(269) goes ON until the diagnostic output is turned ON.
3. If the measured time exceeds the monitoring time setting, a setting 1.5 times the measured time will be stored in T.

Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 01FF or 8000 to 81FF. ON if T is not within the specified range of 0000 to 270F. OFF in all other cases.
Carry Flag	CY	ON if the diagnostic output is still OFF after the monitoring time has elapsed. OFF in all other cases.

The following table shows relevant words and flags in the Auxiliary Area.

Name	Address	Operation
FAL Error Flag	A402.15	ON when a non-fatal (FAL) error is registered in time monitoring.
Executed FAL Number Flags	A360.01 to A391.15	When a non-fatal (FAL) error is registered in time monitoring, the corresponding flag will be turned ON. Flags A360.01 to A391.15 correspond to FAL numbers 0001 to 01FF.
Error Log Area	A100 to A199	The Error Log Area contains the error codes and time/date of occurrence for the most recent 20 errors, including errors generated by FPD(269).

Name	Address	Operation
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.
FPD Teaching Bit	A598.00	Turn this bit ON when you want the monitoring time to be set automatically (teaching function) when FPD(269) is executed.

Precautions

When the time monitoring function is being used, the execution condition for FPD(269) must remain ON for the entire monitoring time set in T.

The execution condition for FPD(269) must be made up of a combination of normally open and normally closed inputs.

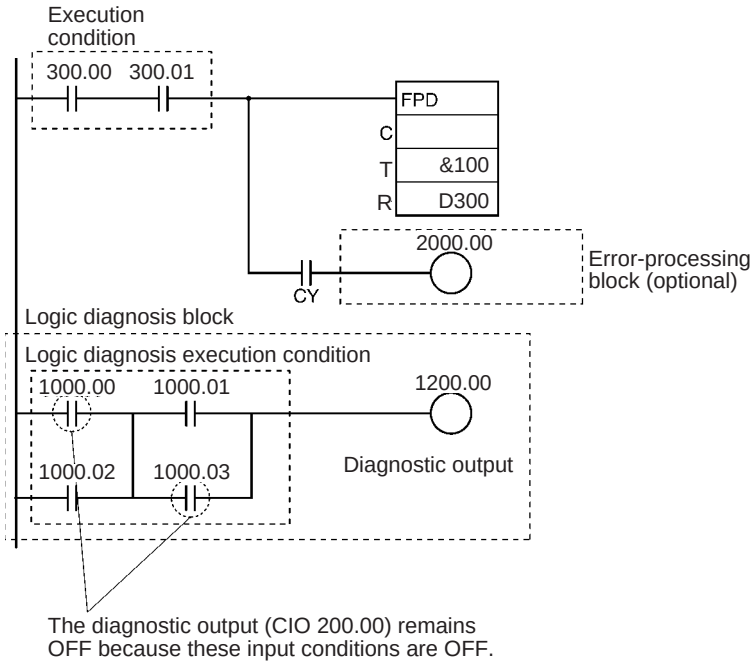
The error-processing block is optional. When an error-processing block is included, be sure to use outputs or other right-hand instructions. LD and LD NOT cannot be used at this point.

FPD(269) can be used more than once in the program, but each instruction must have a unique register (R) setting.

The monitoring time is refreshed only when FPD(269) is executed. If the cycle time is longer than 100 ms, the monitoring time will not be refreshed normally and FPD(269) will not operate correctly because the monitoring time is updated in units of 100 ms.

Examples

The following program example is used to demonstrate the operation of the time monitoring function and logic diagnosis function. In this example, the diagnostic output (CIO 1200.00) does not go ON because CIO 1000.00 and CIO 1000.03 remain OFF in the logic diagnosis execution condition.



Time Monitoring Function

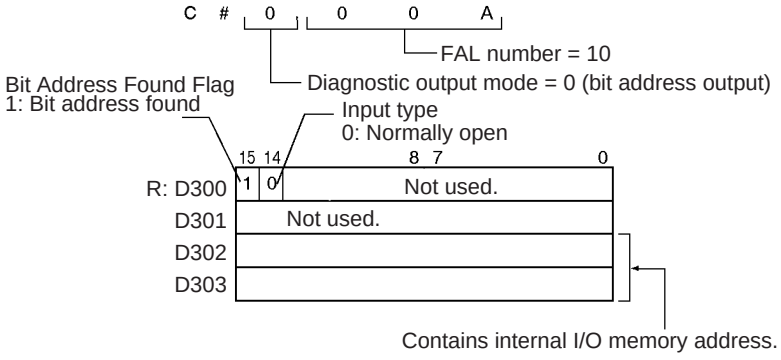
If the diagnostic output (CIO 1200.00) does not go ON within 10 seconds after CIO 300.00 and CIO 300.01 are both ON, a non-fatal error will be generated and the following processing will be performed.

1,2,3...

1. The Carry Flag is turned ON.
2. When the rightmost 3 digits of C specify an FAL number of 00A hex (10), the corresponding Executed FAL Number Flag (A360.10) will be turned ON, the corresponding error code (410A) is written in A400, and the FAL Error Flag (A402.15) is turned ON.

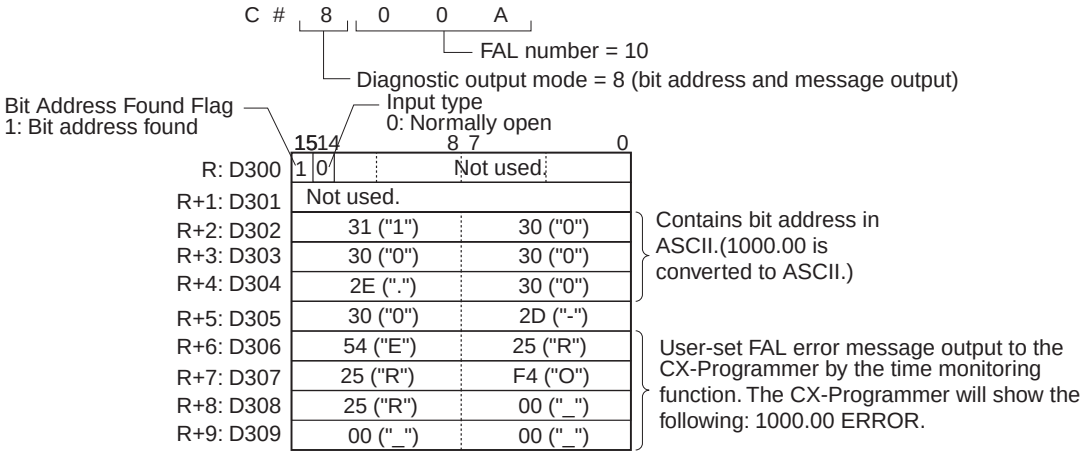
Logic Diagnosis Function (C=000A)

Since the leftmost digit of C is 0 (bit address output mode) the PLC memory address of CIO 1200.00 is output to D303 and D302. (CIO 1000.00 is on a higher instruction line than CIO 1000.03.)



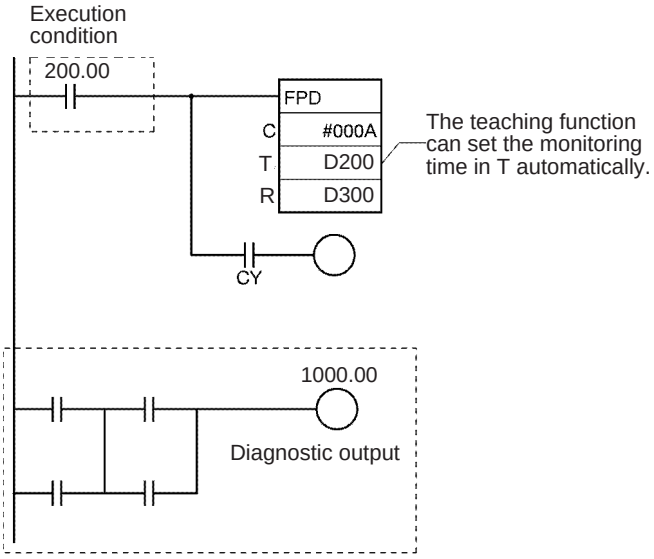
Logic Diagnosis Function (C=800A)

Since the leftmost digit of C is 8 (bit address and message output mode) the address of CIO 1000.00 (1000.03) is output to D302 through D304 in ASCII.

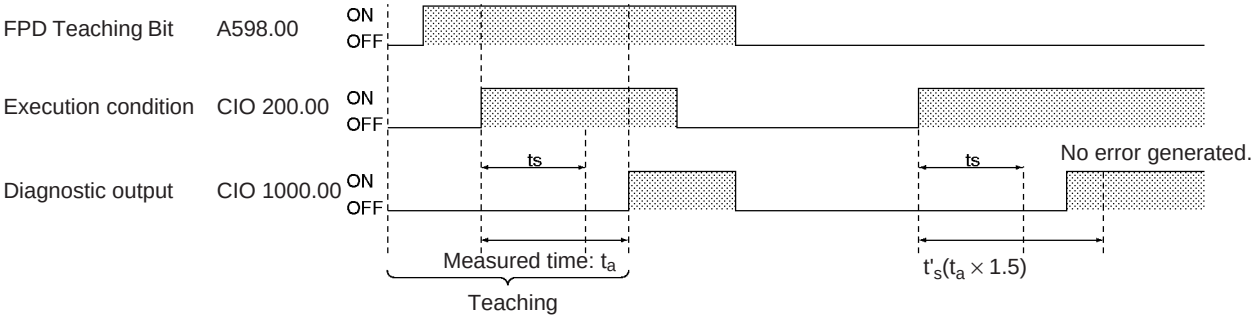


Setting the Monitoring Time with the Teaching Function

The monitoring time can be set automatically with the teaching function when a word address has been specified for T.



To start the teaching function, turn ON A598.00 (the FPD Teaching Bit). While A598.00 is ON, FPD(269) measures how long it takes for the diagnostic output (CIO 200.00) to go ON after the execution condition (CIO 1000.00) goes ON. If the measured time exceeds the monitoring time in T, the measured time is multiplied by 1.5 and that value is stored in T as the new monitoring time.



t_s: Initial setting in T

t_a: Measured time

t'_s: New setting in T after teaching

(When $t_a > t_s$, $t'_s = t_a \times 1.5$)

3-29 Other Instructions

This section describes instructions for manipulating the Carry Flag, extending the maximum cycle time, saving/loading Condition Flag status, and converting memory addresses.

Instruction	Mnemonic	Function code	Page
SET CARRY	STC	040	958
CLEAR CARRY	CLC	041	958
EXTEND MAXIMUM CYCLE TIME	WDT	094	959
SAVE CONDITION FLAGS	CCS	282	961
LOAD CONDITION FLAGS	CCL	283	963
CONVERT ADDRESS FROM CV	FRMCV	284	964
CONVERT ADDRESS TO CV	TOCV	285	968

3-29-1 SET CARRY: STC(040)

Purpose

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.
Immediate Refreshing Specification		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 which affect the Carry Flag.

Flags

Name	Label	Operation
Error Flag	ER	---
Equals Flag	=	---
Carry Flag	CY	ON
Negative Flag	N	---

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.

3-29-2 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.
Immediate Refreshing Specification		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(040) turns OFF the Carry Flag (CY). Although CLC(040) 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	---
Equals Flag	=	---
Carry Flag	CY	OFF
Negative Flag	N	---

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.

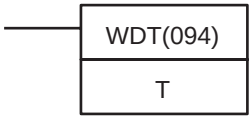
Note The +(400), +L(401), +B(404), +BL(405), –(410), –L(411), –B(414), and –BL(415) instructions do no include the Carry Flag in their addition and subtraction operations. In general, use these instructions when performing addition or subtraction.

3-29-3 EXTEND MAXIMUM CYCLE TIME: WDT(094)

Purpose

Extends the maximum cycle time, but only for the cycle in which the instruction is executed. WDT(094) can be used to prevent errors for long cycle times when a longer cycle time is temporarily required for special processing.

Ladder Symbol



T: Timer setting

Variations

Variations	**Executed Each Cycle for ON Condition**	WDT(094)
	Executed Once for Upward Differentiation	@WDT(094)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

T: Timer Setting

Specifies the watchdog timer setting between 0000 and 0F9F hexadecimal or between &0000 and &3999 decimal.

Operand Specifications

Area	T
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	0000 to 0F9F (binary)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

WDT(094) extends the maximum cycle time for the cycle in which this instruction is executed. The watchdog timer setting in the PLC Setup is extended by an interval of T × 10 ms (0 to 39,990 ms).

The following table shows the watchdog timer settings in the PLC Setup. The default value for the maximum cycle time is 1,000 ms, although it can be set anywhere from 1 to 40,000 ms in 10-ms units.

Name	Function	Settings
Watch cycle time	A Cycle Time Too Long error (fatal error) will be registered if the cycle time exceeds the maximum setting.	0: Default setting (1,000 ms) 1: User time setting
	Sets the maximum cycle time. (This setting is valid only when the first setting has been set to 1.)	0001 to 0FA0 (1 to 40,000 ms, 10-ms units)

Flags

Name	Label	Operation
Error Flag	ER	ON if the watchdog timer setting exceeds 40 seconds. OFF in all other cases.

The following table shows relevant flags and words in the Auxiliary Area.

Name	Address	Operation
Cycle Time Too Long Flag	A401.08	ON when the present cycle time exceeds the maximum cycle time (watch cycle time) set in the PLC Setup. This is a fatal error which causes program execution to stop.

Name	Address	Operation
Maximum Cycle Time	A262 and A263	These words contain the maximum cycle time in 32-bit binary. This value is updated every cycle.
Present Cycle Time	A264 and A265	These words contain the present cycle time in 32-bit binary. This value is updated every cycle.

Precautions

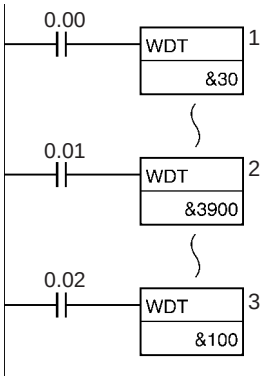
WDT(094) can be used more than once in a cycle. When WDT(094) is executed more than once the cycle time extensions are added together, although the total must not exceed 40,000 ms. If WDT(094) cannot be executed again if the cycle has already been extended to 40,000 ms.

Examples

The default maximum cycle time (1,000 ms) is used in this example.

1,2,3...

1. When CIO 0.00 turns ON, the first WDT(094) instruction extends the maximum cycle time by 300 ms (30 × 10 ms). Thus, the maximum cycle time is 1,300 ms at this point.
2. When CIO 0.01 turns ON, the second WDT(094) instruction attempts to extend the maximum cycle time by another 39,000 ms. Since the new maximum cycle time (40,300 ms) exceeds the upper limit of 40,000 ms, the extra 300 ms is ignored. As a result, the second WDT(094) instruction actually extends the maximum cycle time by 38,700 ms.
3. When CIO 0.02 turns ON, the third WDT(094) instruction attempts to extend the maximum cycle time by another 1,000 ms. Since the maximum cycle time has already reached the upper limit of 40,000 ms, the third WDT(094) instruction is not executed.

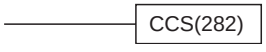


3-29-4 SAVE CONDITION FLAGS: CCS(282)

Purpose

Saves the current status of the Condition Flags in a separate area within the CPU Unit. The current status of the Flags is preserved so that it can be read (restored) with CCL(283) at a different location in the program, in a different task, or even in a later cycle.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	CCS(282)
	Executed Once for Upward Differentiation	@CCS(282)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

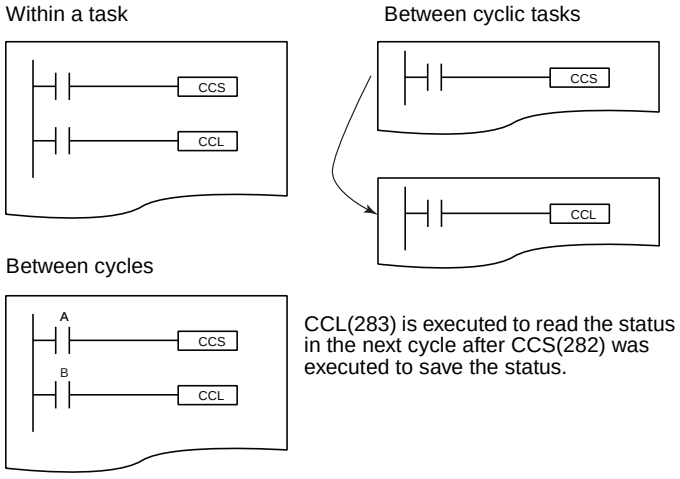
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

When the execution condition is ON, CCS(282) stores the current status of the Condition Flags (except for the ALWAYS ON and ALWAYS OFF Flags) in a separate area in the CPU Unit. The Status of the following Condition Flags will be preserved: ER, CY, >, =, <, N, OF, UF, >=, <>, and <=.

The preserved status of the Condition Flags can be read (restored) later only with CCL(283), the LOAD CONDITION FLAGS instruction. The status can be read in any of the following cases:

- Within a task
- Between different cyclic tasks
- Between cycles



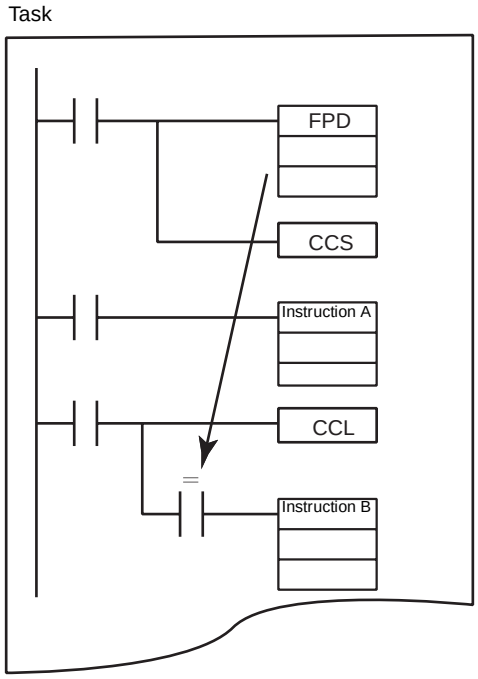
Note

(1) The status of the Condition Flags cannot be saved/loaded between a cyclic task and interrupt task.

(2) When CCS(282) is executed, it overwrites the previous Condition Flag information that was saved.

All of the Condition Flags are cleared when operation switches from one task to another. Use the CCS(282) and CCL(283) instructions to save and load the Condition Flag status between tasks or cycles.

For example, the CCS(282) and CCL(283) instructions make it possible to use the CY Flag status (time monitoring diagnosis error) from the execution of FPD(269) at a later point in the program, not immediately after execution of the instruction.



The results of the comparison are stored in the Condition Flags. (In this case, the results of the COMPARE Instruction can be used in instruction B even if those results are affected by execution of instruction A.)

Preserves the status of the Condition Flags in a separate location in the CPU Unit.

Restores the status of the Condition Flags.

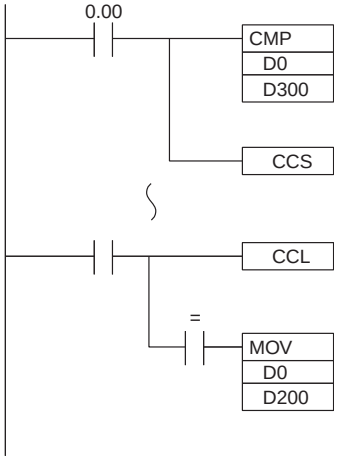
The Equals Flag will reflect the result of the COMPARE instruction, not the result of instruction A.

Flags

There are no flags affected by these instructions.

Examples

In the following example, CCS(282) preserves the results of a Comparison so that this result can be used as an execution condition later in the program.



When CIO 0.00 is ON in the following example, CMP(020) will compare the contents of D0 and D300 and those results will be preserved by CCS(282).

The preserved Condition Flags are restored by CCL(283).

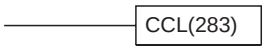
This MOV(021) instruction is executed if the result of the CMP(020) instruction caused the Equals flag to be turned ON.

3-29-5 LOAD CONDITION FLAGS: CCL(283)

Purpose

Restores the status of the Condition Flags that were saved in a separate area within the CPU Unit by CCS(282). It is also possible to use CCL(283) independently to clear the Condition Flags.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	CCL(283)
	Executed Once for Upward Differentiation	@CCL(283)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

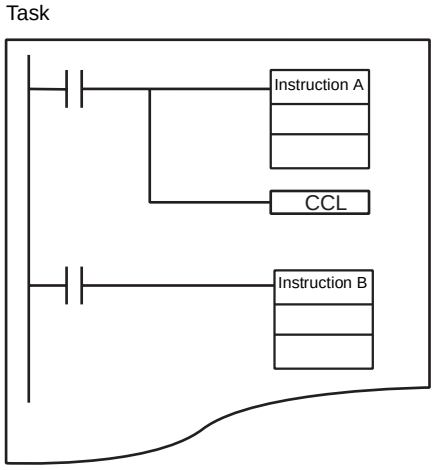
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

When the execution condition is ON, CCL(283) restores (reads) the status of the Condition Flags (except for the ALWAYS ON and ALWAYS OFF Flags). The Status of the following Condition Flags will be restored (read): ER, CY, >, =, <, N, OF, UF, >=, <>, and <=.

Condition Flags are shared by all instructions, so the status of these Flags may change many times during the PLC cycle as each instruction is executed. Previously, it was necessary to place conditions using the Condition Flags immediately after the controlling instruction so that the status of the Condition Flags would not be affected by intervening instructions. The CCS(282) and CCL(283) instructions allow the controlling instruction to be separated from the execution conditions that rely on the result.

For example, CCS(282) can store the status of the Equals Flag after execution of a Comparison Instruction and the result can be restored later. The result does not have to be used immediately after execution of the instruction.



CCL(283) is used alone to clear the Condition Flags after execution of instruction A so that those results do not affect instruction B and later instructions.

Refer to *3-29-4 SAVE CONDITION FLAGS: CCS(282)* for more examples showing how to use CCS(282) and CCL(283).

Flags

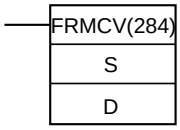
There are no flags affected by these instructions.

3-29-6 CONVERT ADDRESS FROM CV: FRMCV(284)

Purpose

Converts a CV-series PLC memory address to its corresponding CS/CJ/CP-series PLC memory address. FRMCV(284) can be useful when converting CV-series programs that use PLC memory addresses so that they are compatible with CP-series PLCs.

Ladder Symbol



S: Word containing the CV-series PLC memory address
D: Destination Index Register

Variations

Variations		
	Executed Each Cycle for ON Condition	FRMCV(284)
	Executed Once for Upward Differentiation	@FRMCV(284)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

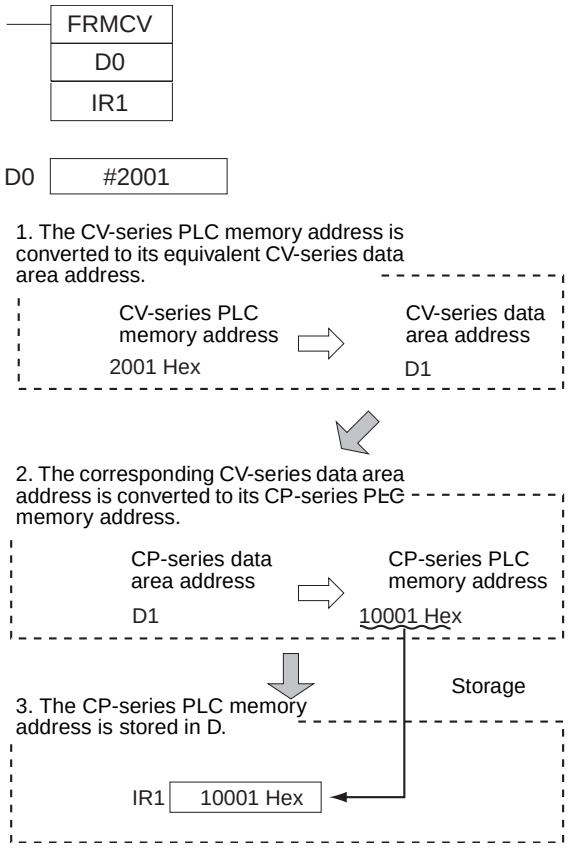
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

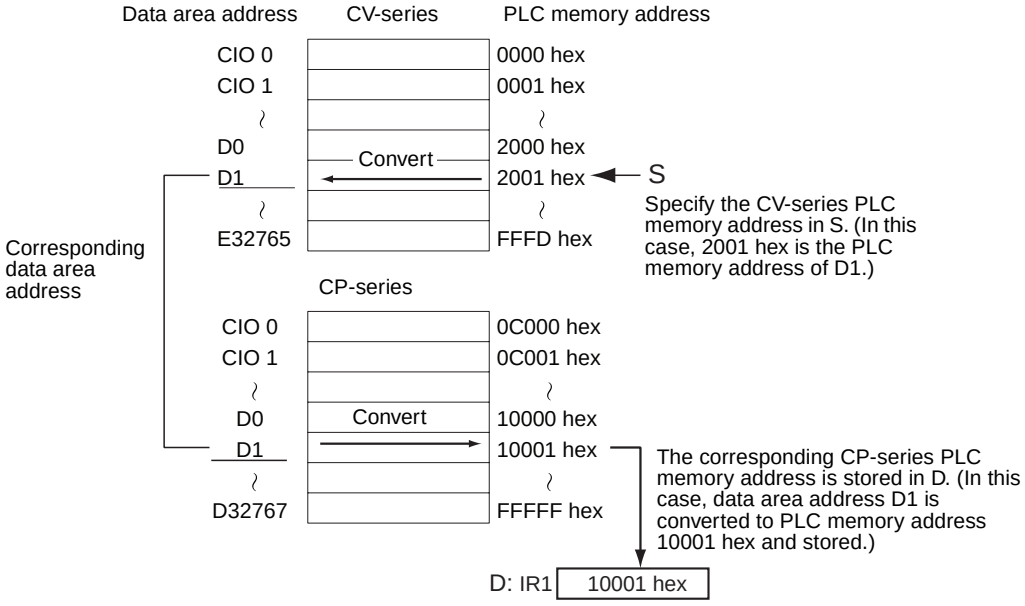
Description

When the execution condition is ON, FRMCV(284) executes the following operations.

1. The CV-series PLC memory address specified in S is converted to its equivalent CV-series data area address.
2. FRMCV(284) determines the CP-series PLC memory address that corresponds to the same CV-series data area address.
3. The CP-series PLC memory address is output to D. (An index register (IR0 to IR15) must be specified for D.)

The following example shows FRMCV(284) used to convert the CV-series PLC memory address in D0.





Note If there is no CP-series equivalent to the specified CV-series PLC memory address, an error will occur, the Error Flag will be turned ON, and the address will not be converted.

When an Index Register is used as an operand with a ",IR" prefix, the instruction will operate on the word indicated by the PLC memory address in the Index Register, not the Index Register itself. Once the desired PLC memory address has been stored in an Index Register, the Index Register itself can be used as an operand for an instruction.

The FRMCV(284) instruction can be used to convert a CV-series program with the following two kinds of programming for use in a CP-series PLC. See the *Examples* later in this section for an example.

1. When using indirect binary mode DM addressing (*DM)
   (when indirectly specifying a data area address with a PLC memory address in DM)
2. When using CV-series PLC memory addresses directly as values
   (when storing PLC memory addresses in Index Registers with direct addressing using an instruction such as MOV(021))

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6143	---
Work Area	W0 to W511	---
Holding Bit Area	H0 to H511	---
Auxiliary Bit Area	A448 to A959	---
Timer Area	T0000 to T4095	---
Counter Area	C0000 to C4095	---
DM Area	D0 to D32767	---
Indirect DM addresses in binary	@ D0 to @ D32767	---
Indirect DM addresses in BCD	*D0 to *D32767	---
Constants	Any constant except 09FF hex, 0A00 to 0AFF hex, or 0D00 to 0E3F hex	---
Data Registers	DR0 to DR15	---

Area	S	D
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 IR15	---

Flags

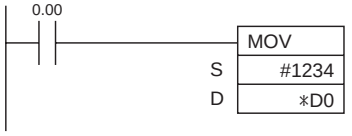
Name	Label	Operation
Error Flag	ER	ON if S specifies one of the following PLC memory addresses that do not exist in the CP-series: Temporary Relay (TR) Area (09FF hex) CPU Bus Link (G) Area (0A00 to 0AFF hex) SFC Areas (0D00 to 0E3F hex) OFF in all other cases.

Examples

Example 1: Converting a CV-series Program with ∗DM Indirect Binary Mode DM Addressing

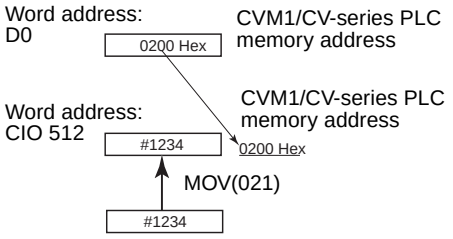
In this FRMCV(284) example, a DM word is specified in S, the PLC memory address there is stored in an Index Register, and the Index Register is used for indirectly addressed.

• CV-series program
(Program using indirect DM binary mode addressing)



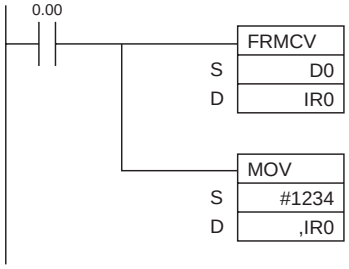
PLC Setup
Indirect DM data:
When indirect DM addresses are in binary, the content of the DM word is treated as a PLC memory address and specifies the corresponding address in I/O memory.

In this case, the value in D0 is 200 hex. The corresponding data area address is CIO 512, so #1234 is transferred to CIO 512.

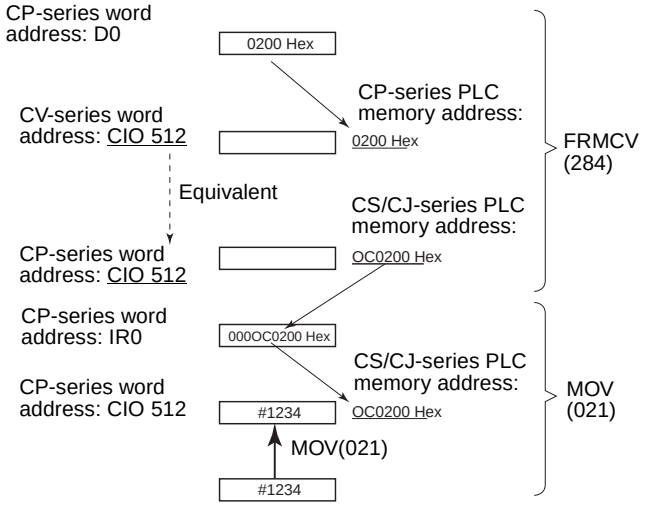


Equivalent program

• CP-series program



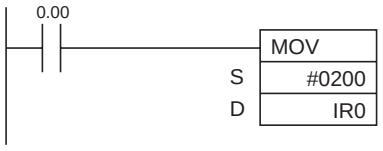
In this case, the value in D0 is 200 hex. The corresponding CV-series data area address is CIO 512. The CP-series PLC memory address for CIO 512 is 0000C200 hex, so this value is stored in IR0. The destination operand in MOV(021) indirectly addresses the content of IR0, so #1234 is transferred to CIO 512.



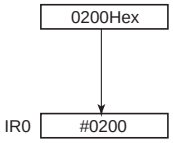
Example 2: Converting a CV-series Program with PLC Memory Addresses Stored directly in Index Registers

In this FRMCV(284) example, the CV-series PLC memory address is specified directly in S.

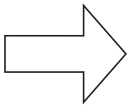
• CV-series program
(Program using PLC memory addresses stored directly in IR)



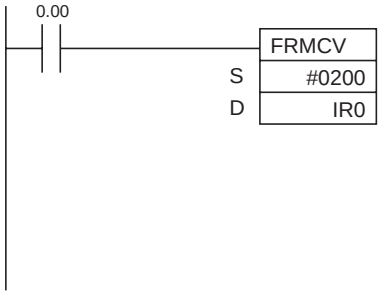
In this case, the PLC memory address 0200 hex is stored in Index Register IR0.



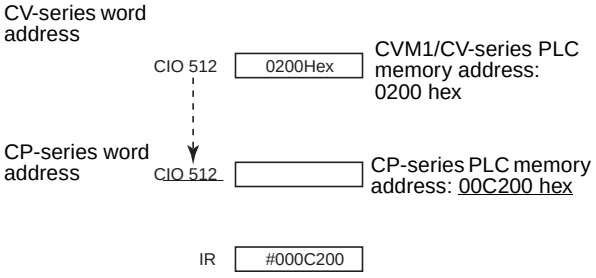
Equivalent program



• CP-series program



In this case, the CV-series PLC memory address 0200 hex corresponds to CIO 512. The CP-series PLC memory address for CIO 512 is 0000C200 hex, so this value is stored in IR0.

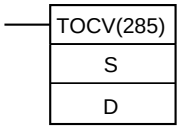


3-29-7 CONVERT ADDRESS TO CV: TOCV(285)

Purpose

Converts a CS/CJ/CP-series PLC memory address to its corresponding CV-series PLC memory address. TOCV(285) can be useful when converting CP-series programs that use PLC memory addresses so that they are compatible with CV-series PLCs.

Ladder Symbol



S: Index Register containing the CP-series PLC memory address

D: Destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	TOCV(285)
	Executed Once for Upward Differentiation	@TOCV(285)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

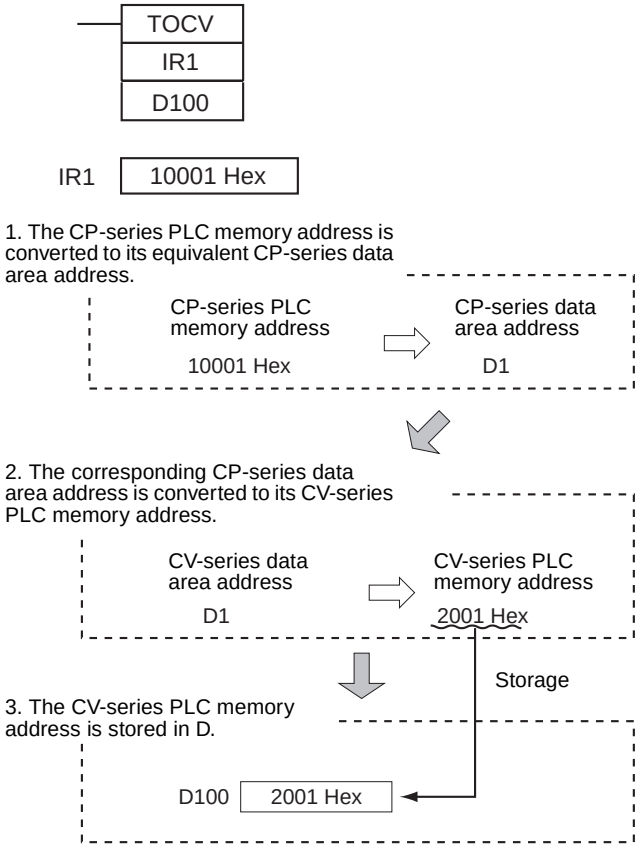
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

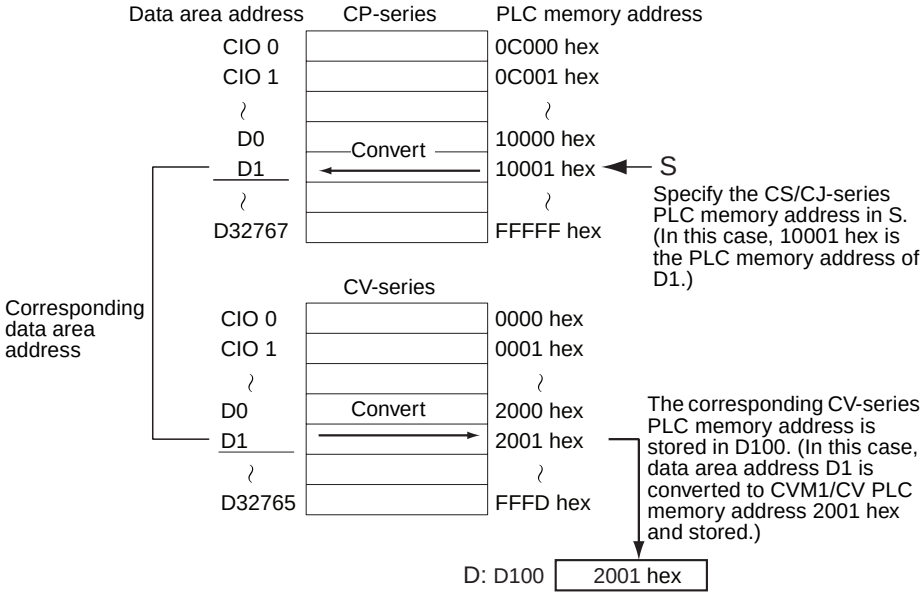
Description

When the execution condition is ON, TOCV(285) executes the following operations.

1. The CP-series PLC memory address specified in S is converted to its equivalent CP-series data area address. (An index register (IR0 to IR15) must be specified for S.)
2. TOCV(284) determines the CV-series PLC memory address that corresponds to the same CP-series data area address.
3. The CV-series PLC memory address is output to D.

The following example shows TOCV(285) used to convert the PLC memory address in IR1.





Note

(1) If there is no CV-series equivalent to the specified CP-series PLC memory address, an error will occur, the Error Flag will be turned ON, and the address will not be converted.

(2) The CV-series PLC memory address data stored by TOCV(285) can be transferred to a CV-series PLC using CX-Programmer.

(3) The same data area address that was used in the CP-series program can be specified in the CV-series program by using indirect Index Register addressing (",IR" prefix) or indirect binary mode DM addressing (*DM).

Operand Specifications

Area	S	D
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A448 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	See note 1.	---
Data Registers	---	DR0 to DR15
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 IR15

Note (1) An error will occur and the Error Flag will be turned ON if S specifies one of the following PLC memory addresses that do not exist in the CV-series:

Area or addresses	PLC memory addresses
Task Flag Area	0000 B800 to 0000 B801 hex
A512 to A959	0000 BA40 to 0000 BBFF hex
CIO 2556 to CIO 6143	0000 C9FC to 0000 D7FF hex
T1024 to T4095	0000 BE40 to 0000 BEFF hex and 0000 E400 to 0000 EFFF hex
C1024 to C4095	0000 BF40 to 0000 BFFF hex and 0000 F400 to 0000 FFFF hex
HR Area	0000 D800 to 0000 D9FF hex
WR Area	0000 DE00 to 0000 DFFF hex
D24576 to D32767	0001 6000 to 0001 7FFF hex

(2) An error will occur and the Error Flag will be turned ON if an area other than the Index Register Area is specified for S.

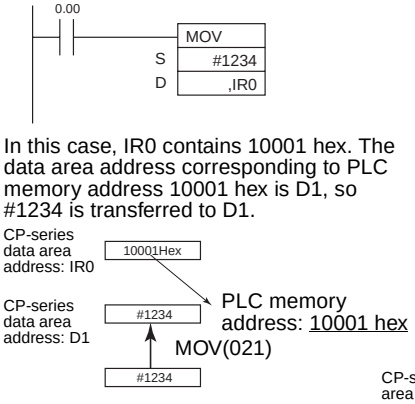
Flags

Name	Label	Operation
Error Flag	ER	ON if S specifies a PLC memory address that does not exist in the CV-series PLCs. ON if S is not a constant or Index Register. OFF in all other cases.

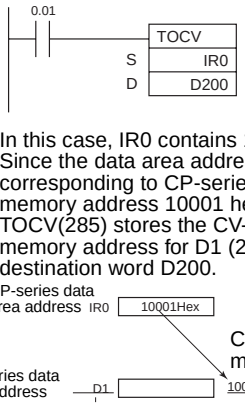
Example **Converting a CP-series Program with Indirect Index Register Addressing**

1. In this TOCV(285) example, an Index Register is specified in S. The CP-series PLC memory address in that Index Register is converted to its CV-series equivalent.
2. The CV-series PLC memory address is transferred to the specified data area address.
3. Use the CV-series PLC memory address in the CV-series program.

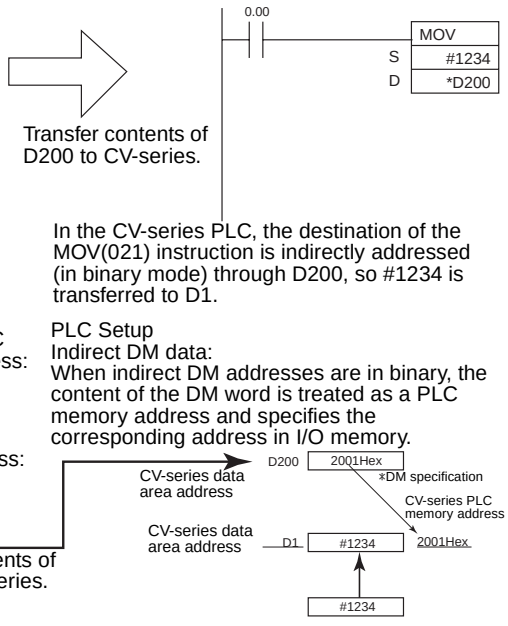
• CP-series program (Program using indirect Index Register addressing)



• CP-series program



• CV-series program



3-30 Block Programming Instructions

This section describes block programs and the block programming instructions.

Instruction	Mnemonic	Function code	Page
BLOCK PROGRAM BEGIN	BPRG	096	976
BLOCK PROGRAM END	BEND	801	976
BLOCK PROGRAM PAUSE	BPPS	811	979
BLOCK PROGRAM RESTART	BPRS	812	979
CONDITIONAL BLOCK EXIT (NOT)	EXIT (NOT)	806	985
IF (NOT)	IF (NOT)	802	981
ELSE	ELSE	803	981
IF END	IEND	804	981
ONE CYCLE AND WAIT (NOT)	WAIT (NOT)	805	988
TIMER WAIT	TIMW (BCD)	813	992
	TIMWX (binary)	816	
COUNTER WAIT	CNTW (BCD)	814	995
	CNTWX (binary)	818	
HIGH-SPEED TIMER WAIT	TMHW (BCD)	817	998
	TMHWX (binary)	815	
LOOP	LOOP	809	1001
LOOP END (NOT)	LEND (NOT)	810	1001

3-30-1 Introduction

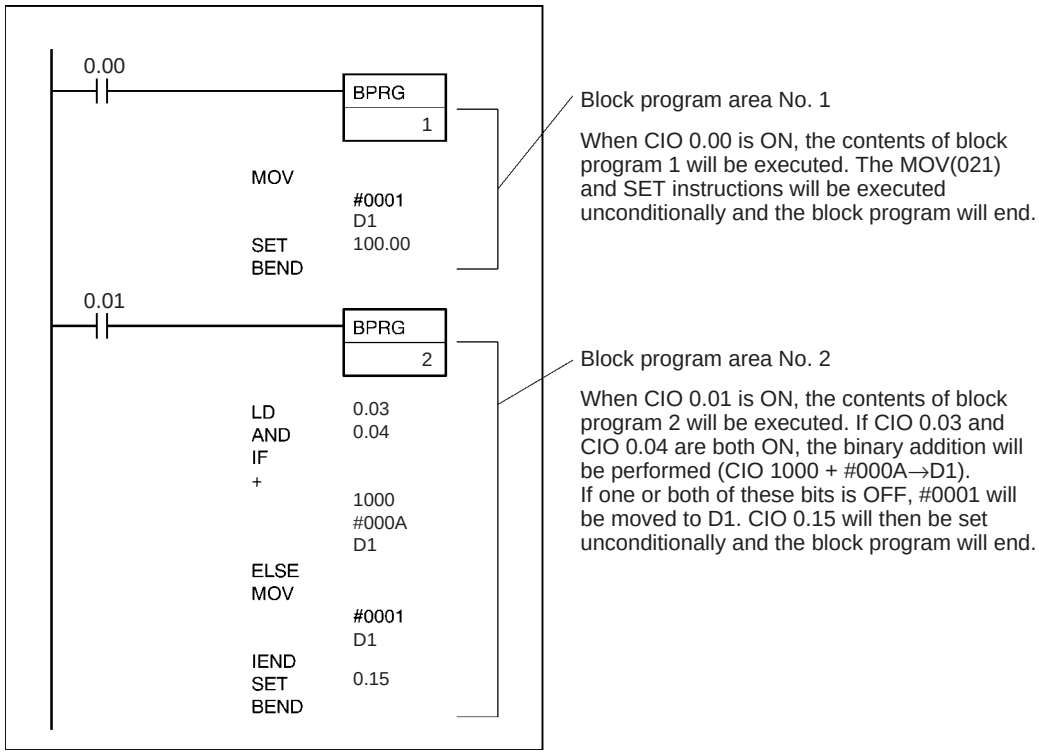
Block Programs

Up to 128 block programs can be used 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 allow 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.



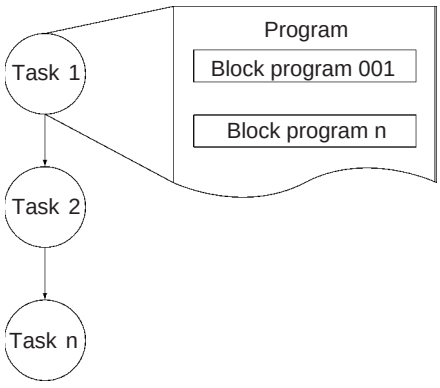
Tasks and Block Programs

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 that are not executed (i.e., which have an OFF execution 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 with block programs unless intentionally programmed with IF(802), WAIT(805), EXIT(806), IEND(810) or other instructions. Also, there are some instructions that cannot be used within block programs, such as those that detect upward and downward differentiation.

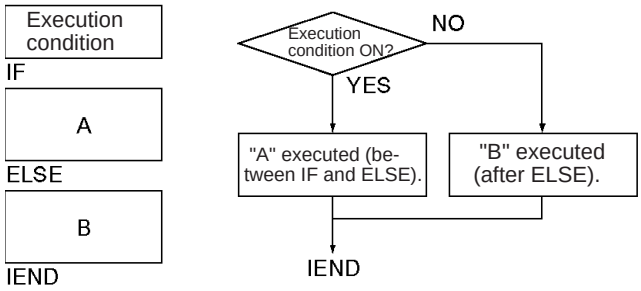
Block programs can be used either within cyclic tasks or interrupt tasks. Each block program number from 0 to 127 can be used only once and cannot be use again, even in a different task.



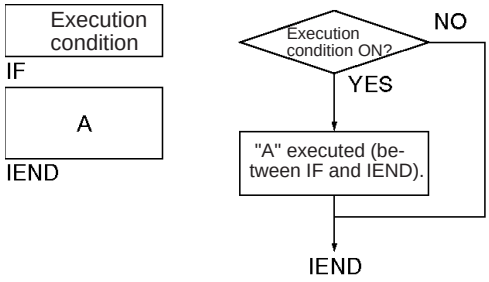
Using Block Programming Instructions

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.



If execution is to wait until an execution condition or bit is ON (e.g., for step progressions), then WAIT(805) is used.

If execution is to wait until for a specified period of time (e.g., for timed step progressions), then TIMW(813), TIMX(816), TMHW(815), or TMHWX(817) is used.

If execution is to wait until for a specified count has been reached (e.g., for step progressions with counters), then CNTW(814)/CNTWX(818) is used.

If execution is to be repeated within part of a block program until a condition is met, then LOOP(809) and LEND(810) are used.

If execution of the block program is to be ended in the middle based on an execution condition, the EXIT(806) is used.

If another block program that is being executed is to be paused or restarted from within a block program, then BPPS(811) and BPRS(812) are used.

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(802) (NOT)
	ONE CYCLE AND WAIT (NOT)	WAIT(805) (NOT)
	EXIT	EXIT(806) NOT
	LOOP END	LEND(810) NOT
Ladder diagram instructions	CONDITIONAL JUMP	CJP(510)
	CONDITIONAL JUMP NOT	CJPN(511)

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), WAIT(805), EXIT(806), LEND(810), CJP(510, or CJPN(511) 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
UP/DOWN	CONDITION ON/CONDITION OFF
>, <,=, >=, <=, <> (S) (L)	Symbol Comparison Instruction (not right-hand instructions)
LD TST/TST NOT	LOAD Bit Test Instructions
AND TST/TST NOT	AND Bit Test Instructions
OR TST/TST NOT	OR Bit Test Instructions
>$, <$,=$, >=$, <=$, <>$	Text String Comparison Instruction

Good Example

```
LD    0.00
AND   0.01
TST   D0     #0010   ] Used as execution condition for IF.
IF
```

Bad Example

```
LD    0.00
AND   0.01
TST   D0     #0010   ] Cannot be used as execution condition for MOV(021).
MOV   #0000  0010
```

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

Instruction group	Mnemonic	Name	Alternative
Sequence Control Instructions	FOR(512) and NEXT(513)	FOR-NEXT LOOPS	Use LOOP(809) and LEND(810) (NOT).
	BREAK(514)	BREAK LOOP	
	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR	Divide the block program into smaller blocks.
	JMP(004)0 and JME(005) 0	Multiple JUMP and Multiple JUMP END	Use JMP(004 and JME(005) (but the jump will be made unconditionally).
	END(001)	END	Use BEND(801).
Timer and Counter Instructions	TIM	TIMER	Use TIMW(813), TIMWX(816), TMHW(815), TMHWX(817), CNTW(814), and CNTWX(818). Other instructions in the block program will not be executed until the timer times out or the counter counts out.
	TIMH(015)	HIGH-SPEED TIMER	
	TMHH(540)	ONE-MS TIMER	
	TTIM(087)	ACCUMULATIVE TIMER	
	TIML(542)	LONG TIMER	
	MTIM(543)	MULTI-OUTPUT 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.
Step Instructions	STEP(008) and SNXT(009)	STEP and STEP NEXT	Use WAIT(805).
Data Control Instructions	PID(190)	PID CONTROL	None
Diagnostic Instructions	FPD(269)	FAILURE POINT DETECTION	None
Upward and Downward Differentiated Instructions	Mnemonics with @	Upward Differentiated Instructions	None
	Mnemonics with %	Downward Differentiated Instructions	None

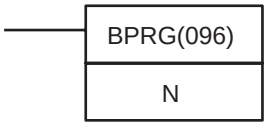
3-30-2 BLOCK PROGRAM BEGIN/END: BPRG(096)/BEND(801)

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		
	Executed Each Cycle for ON Condition	BPRG(096)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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

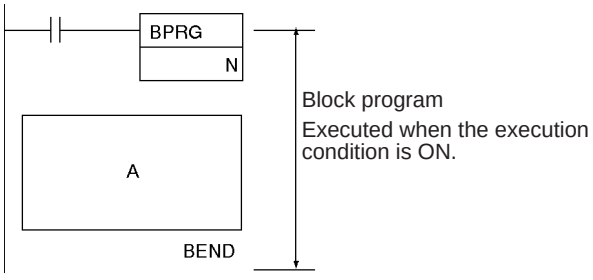
The block program number must be between 0 and 127 decimal.

Operand Specifications (BPRG(096))

Area	N
CIO Area	---
Work Area	---
Holding Bit 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)
Data Registers	---
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 instruction in the block program.

Execution of the block program can be stopped using BPPS(811) from within another block program even if the execution condition for BPRG(096) is ON.

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. ON if the same block program number is used more than once. OFF in all other cases.

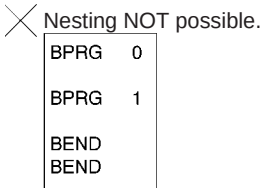
BEND(801)

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.



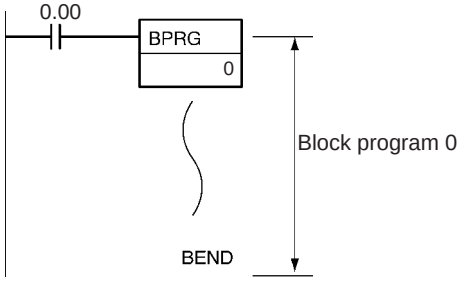
If the block program is in an interlocked program section and the execution condition for IL(002) is OFF, the block program will not be executed.

BPRG(096) and the corresponding BEND(801) must be in the same task.

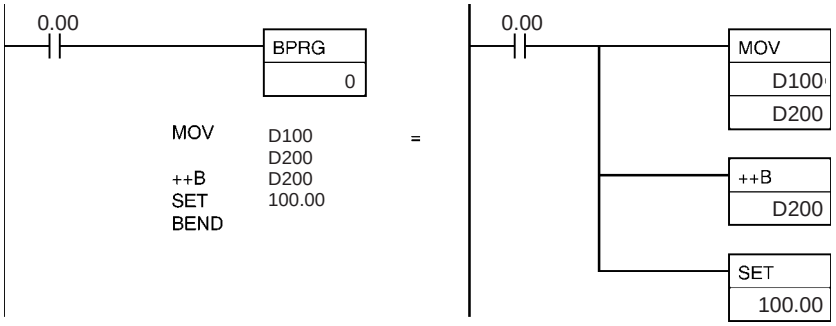
An error will occur and the Error Flag will turn ON if BPRG(096) is in the middle of a block program, BEND(801) is not in a block program, N is not between #0000 and #007F (binary), there is no block program, or if the same block program number is used more than once.

Examples

When CIO 0.00 turns ON in the following example, block program 0 will be executed. When CIO 0.00 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 0.00 turns ON).



3-30-3 BLOCK PROGRAM PAUSE/RESTART: BPPS(811)/BPRS(812)

Purpose

Pause and restart the specified block program from another block program.

Ladder Symbol

BPPS(811) N **N**: Block program number

BPRS(812) N

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

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

Note BPRG(096) and BPRS(812) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

N: Block Program Number

The block program number must be between 0 and 127 decimal.

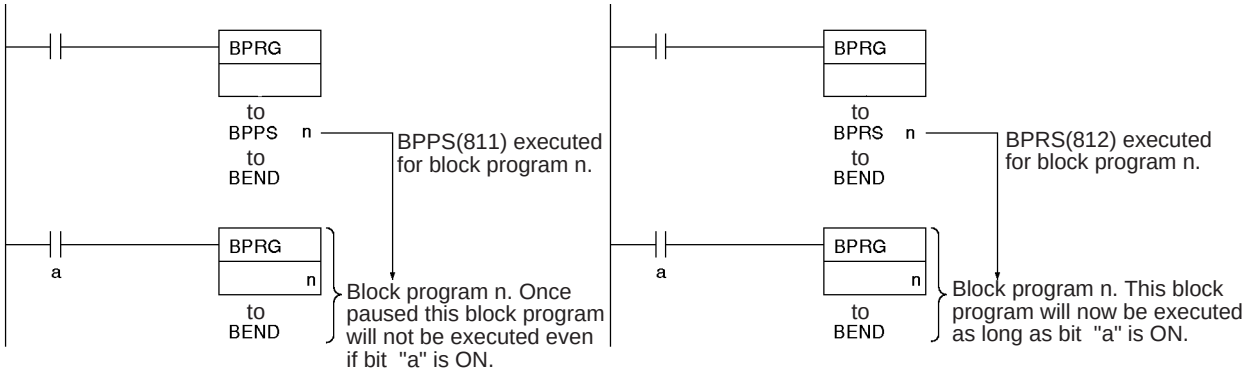
Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit 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)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

BPPS(811) is used inside one block program to pause the execution of another block program specified by N, the block program number. The block program that is paused with BPPS(811) even if the BPRG(096) for the block program has an ON execution condition. The block program will not be restarted until BPRS(812) is executed for it.

BPRS(812) restarts the block program specified by N, the block program number. Once restarted, the block program will be executed as long as the BPRG(096) for the block program has an ON execution condition.



Flags

Name	Label	Operation
Error Flag	ER	ON if BPPS(811) or BPRS(812) is not in a block program. ON if N is not between 0 and 127. OFF in all other cases.

Precautions

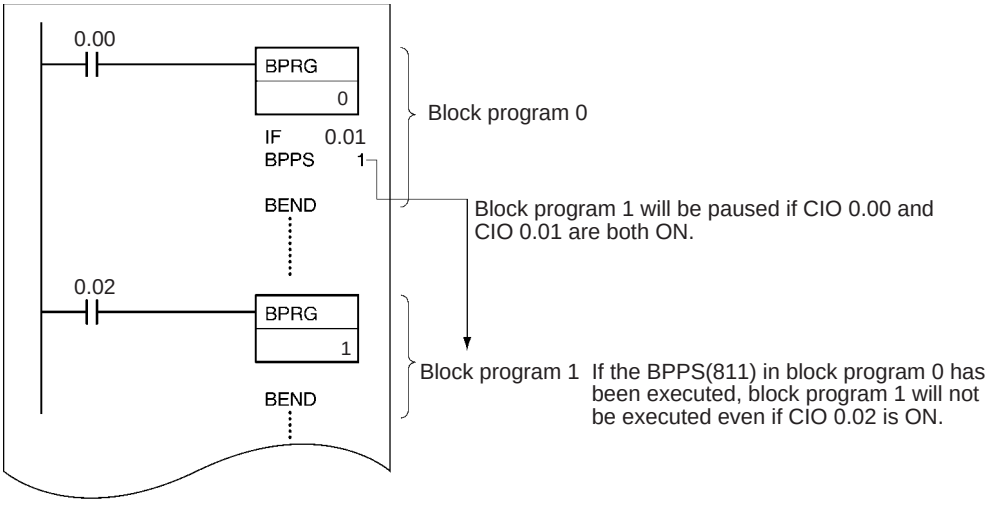
An error will occur and the Error Flag will turn ON if BPPS(811) or BPRS(812) is not in a block program or if N is not between #0000 and #007F (binary).

BPPS(811) can be used to pause the block program that contains it. When the block program is then restarted using BPRS(812) from another block program, the paused block program will restart from the next instruction after BPPS(811).

If a paused block program contains TIMW(813), TIMWX(816), TMHW(815), or TMHWX(817), the PV of the time will continue to elapse even while the block program is paused.

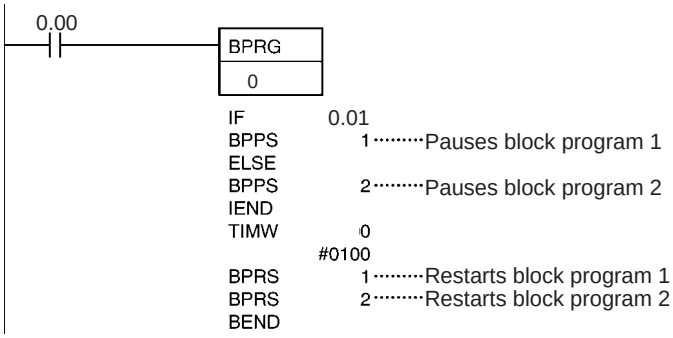
Examples

The following diagram shows a basic example of pausing a block program.



Note If the block program that is being paused appears after BPPS(811), it will not be executed. If the block program appears before BPPS(811), it will be paused starting the next cycle.

If CIO 0.00 is ON, the following program pauses execution of either block program 1 or block program 2 depending on the status of CIO 0.01. The block program that was paused is then restarted after 10 seconds.



Address	Instruction	Operands
000000	LD	0.00
000001	BPRG(096)	00
000002	IF(802)	0.01
000003	BPPS(811)	01
000004	ELSE(803)	
000005	BPPS(811)	02
000006	IEND(804)	
000007	TIMW(803)	0
		# 0100
000008	BPRS(812)	1
000009	BPRS(812)	2
000010	BEND(801)	

3-30-4 Branching: IF(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) must be used in block programming regions even within subroutines and interrupt tasks.

Operand Specifications

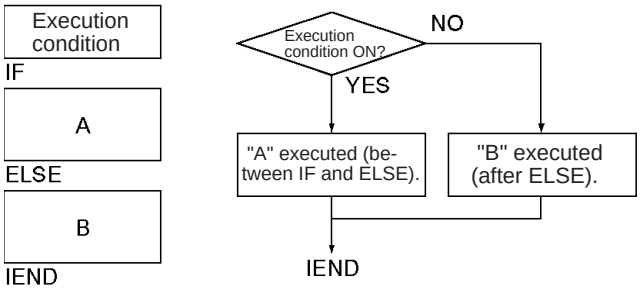
Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A447.15 A448.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK00 to TK31
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON, OFF, AER

Area	B
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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

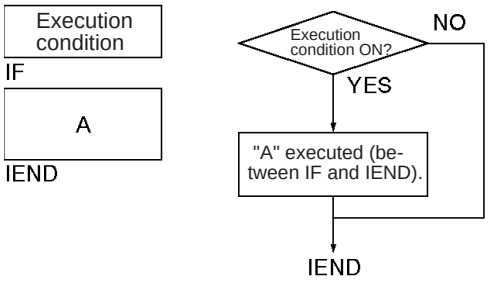
Description

Operation without an Operand for IF(802)

If an operand bit is not specified, an execution 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.

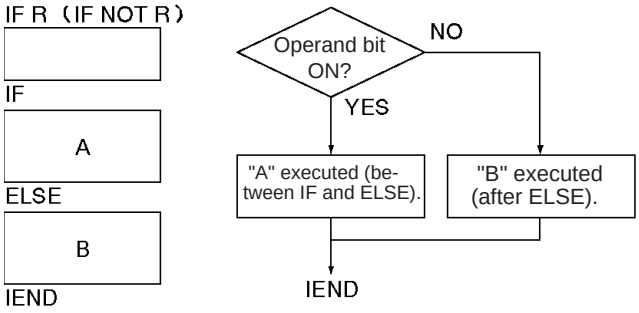


If the ELSE(803) instruction is omitted and the execution condition is ON, the instructions between IF(802) and IEND(804) will be executed and if the execution condition is OFF, only the instructions after IEND(804) will be executed.

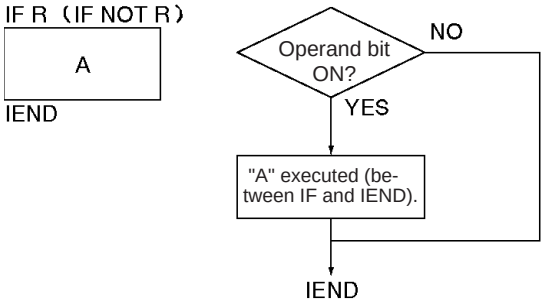


Operation with an Operand for IF(802) or IF NOT(802)

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 and if the operand bit is ON, the instructions be ELSE(803) and IEND(804) will be executed is the operand bit is OFF.



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 254 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.

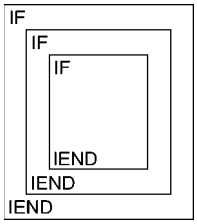
Use IF A IEND to branch between A and doing nothing.

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 254 branches are nested.

Nesting Branches

Up to 253 branches can be nested within the top level branch.

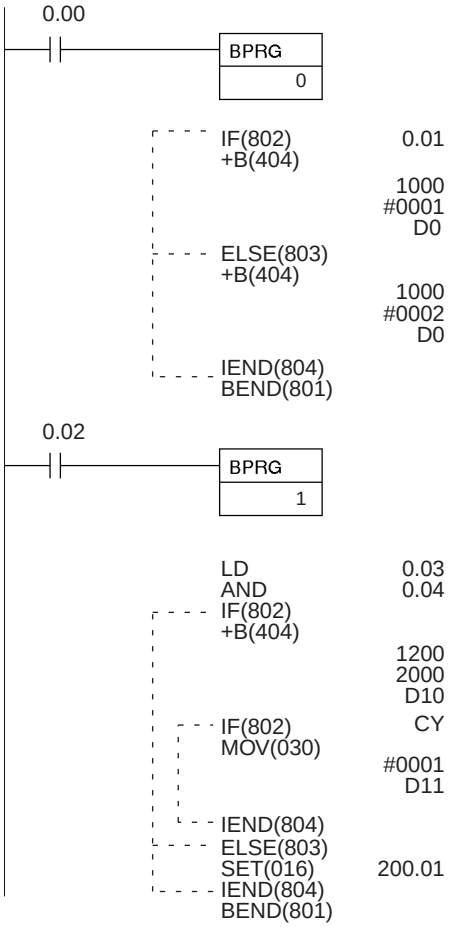


Examples

The following example shows two different block programs controlled by CIO 0.00 and CIO 0.02.

The first block executes one of two additions depending on the status of CIO 0.01. This block is executed when CIO 0.00 is ON. If CIO 0.01 is ON, 0001 is added to the contents of CIO 1000. If CIO 0.01 is OFF, 0002 is added to the contents of CIO 1000. In either case, the result is placed in D0.

The second block is executed when CIO 0.02 is ON and shows nesting two levels. If CIO 0.03 and CIO 0.04 are both ON, the contents of CIO 1200 and CIO 2000 are added and the result is placed in D10 and then 0001 is moved into D11 based on the status of CY. If either CIO 0.03 or CIO 0.04 is OFF, then the entire addition operation is skipped and CIO 200.01 is turned ON.



Address	Instruction	Operands
000000	LD	0.00
000001	BPRG(096)	0
000002	IF(802)	0.01
000003	+B(404)	
		1000
		#0001
		D0
000004	ELSE(803)	
000005	+B(404)	
		1000
		#0002
		D0
000006	IEND(804)	
000007	BEND(801)	
000008	LD	0.02
000009	BPRG(096)	1
000010	LD	0.03
000011	AND	0.04
000012	IF(802)	
000013	+B(404)	
		1200
		2000
		D10
000014	IF(802)	CY
000015	MOV(030)	
		#0001
		D11
000016	IEND(804)	
000017	ELSE(803)	
000018	SET(016)	200.01
000019	IEND(804)	
000020	BEND(801)	

3-30-5 CONDITIONAL BLOCK EXIT (NOT): EXIT (NOT)(806)

Purpose

Exists the block program (i.e., does not execute any other instruction in the block program through BEND(801) depending on the status of the operand bit or on the execution condition. EXIT(806) without an operand bit exits the program if the execution condition is ON. EXIT(806) with an operand bit exits the program if the bit is ON. EXIT NOT(806) must have an operand bit and exits the program if the bit is OFF.

Ladder Symbol

EXIT(806)

EXIT(806) B

EXIT NOT(806) B

B: Bit operand

Variations

Variations	Always Executed in Block Program	EXIT(806)	
		EXIT(806)	B
		EXIT NOT(806)	B

Applicable Program Areas

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

Note EXIT(806) and EXIT NOT(806) must be used in block programming regions even within subroutines and interrupt tasks.

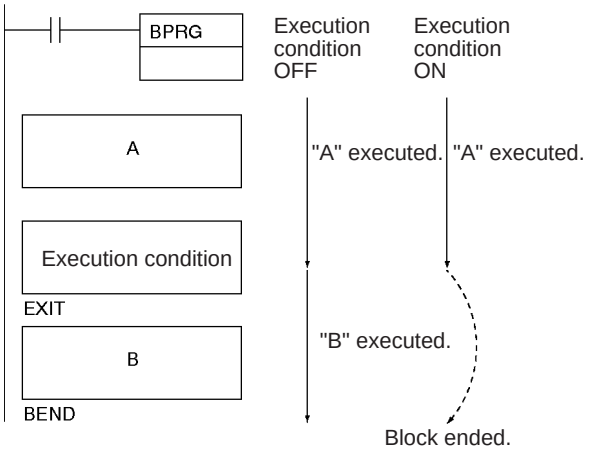
Operand Specifications

Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A447.15 A448.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK00 to TK31
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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

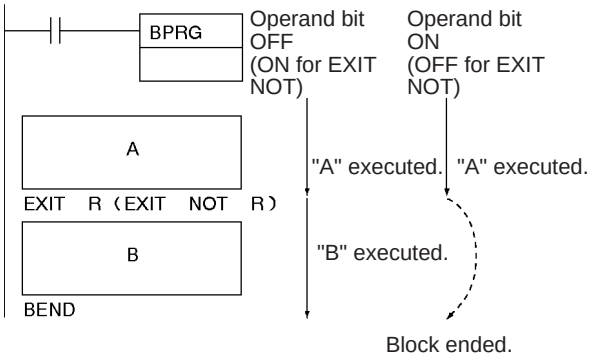
Operation without an Operand

EXIT(806) can be executed without an operand. If it is, then an execution condition must be created for it starting with LD. If the execution condition is OFF, the rest of the block program will be executed normally. If the execution condition is ON, the rest of the instructions in the block program through BEND(801) will not be executed.



Operation with an Operand

If the operand bit, B, is OFF for EXIT(806) the rest of the block program will be executed normally. If the operand bit is ON for EXIT(806), the rest of the instructions in the block program through BEND(801) will not be executed. For EXIT NOT(806), the rest of the block program will be executed for if the operand bit is ON and skipped if the operand bit is OFF.



Flags

Name	Label	Operation
Error Flag	ER	ON if EXIT(806) or EXIT NOT(806) is not in a block program. OFF in all other cases.

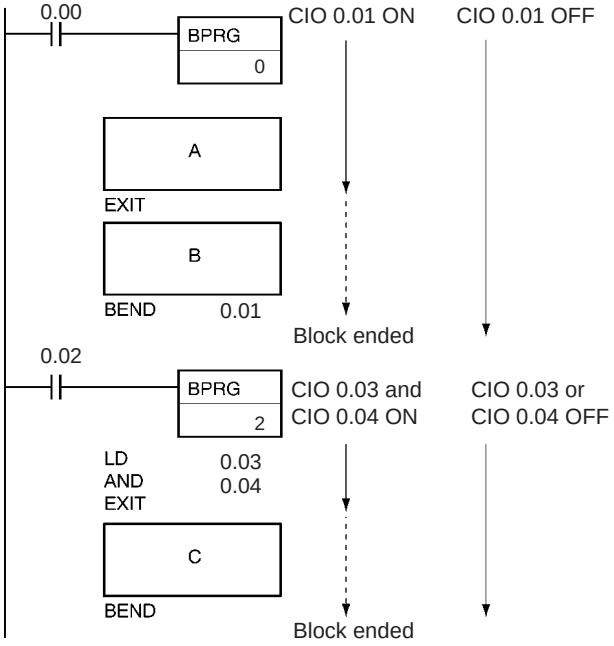
Precautions

An error will occur and the Error Flag will turn ON if EXIT(806) or EXIT NOT(806) is not in a block program.

Examples

When CIO 0.00 is OFF, the block program is executed. If CIO 0.01 is ON, A is executed and then B is skipped and program control jumps to BEND(801). Section B of the program will continue to be skipped until CIO 0.01 turns OFF again.

Although EXIT (NOT)(806) is similar to IF-IEND programming, execution time is normally shorter for EXIT (NOT)(806) because the instructions from EXIT (NOT)(806) to the end of the block program are not executed at all.



Address	Instruction	Operands
000200	LD	0.00
000201	BPRG	0
:	A	:
000210	EXIT	0.01
:	B	:
000220	BEND	---
000221	LD	0.02
000222	BPRG	2
000223	LD	0.03
000224	AND	0.04
000225	EXIT	---
:	C	:
000230	BEND	---

3-30-6 ONE CYCLE AND WAIT (NOT): WAIT(805)/WAIT(805) NOT

Purpose

Stops execution of the rest of the block program until an execution condition turns ON or an operand bit turns ON or OFF.

Ladder Symbol

WAIT(805)

WAIT(805) B

WAIT(805) NOT B

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 WAIT(805)/WAIT(805) NOT must be used in block programming regions even within subroutines and interrupt tasks.

Operand Specifications

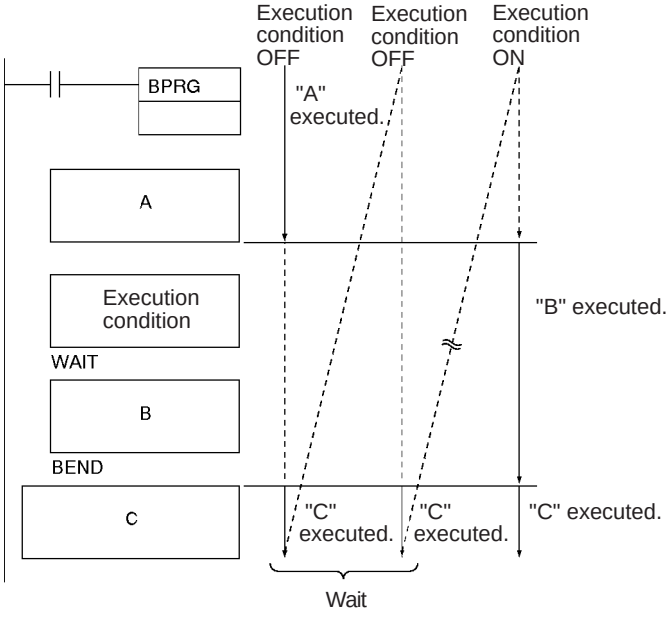
Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A447.15 A448.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095

Area	B
Task Flags	TK00 to TK31
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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

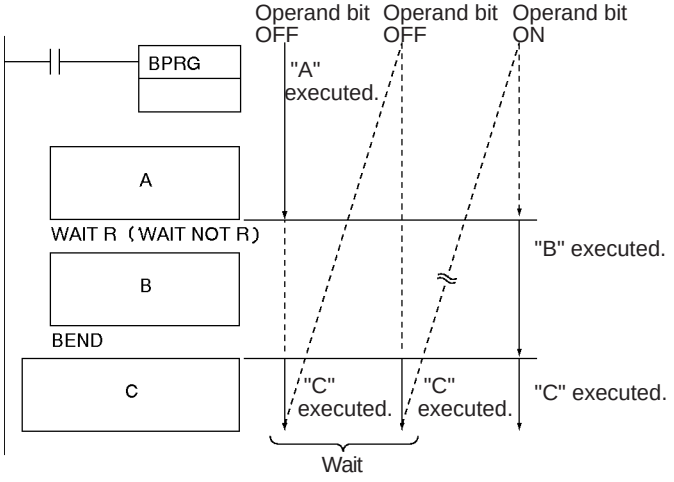
Operation without an Operand

If an operand bit is not specified, an execution must be created before WAIT(805)/WAIT(805 NOT starting with LD. If the execution condition is ON for WAIT(805), the rest of the instruction in the block program will be skipped. In the next cycle, none of the block program will be executed except for the execution condition for WAIT(805). When the execution condition goes ON, the instruction from WAIT(805) to the end of the program will be executed.



Operation with an Operand

An operand bit, B, can be specified for WAIT(805) or WAIT NOT(805). If the operand bit is OFF (ON for WAIT NOT(805)), the rest of the instructions in the block program will be skipped. In the next cycle, none of the block program will be executed except for the execution condition for WAIT(805) or WAIT(805) NOT. When the execution condition goes ON (OFF for WAIT(805) NOT), the instruction from WAIT(805) or WAIT(805) NOT to the end of the program will be executed.



Flags

Name	Label	Operation
Error Flag	ER	ON if WAIT(805) or WAIT(805) NOT is not in a block program. OFF in all other cases.

Precautions

WAIT(805) and WAIT(805) NOT can be used for step progressions inside block programs.

An error will occur and the Error Flag will turn ON if WAIT(805) or WAIT(805) NOT is not in a block program.

Note The program addresses of WAIT instructions with operands specified and the program addresses of the first instruction creating the execution conditions for WAIT instructions without operands are recorded in memory to enable execution to be continued based on the execution condition/bit operand. If online editing performed from the CX-Programmer, however, the WAIT status will be cleared and the block program will again be executed from the beginning.

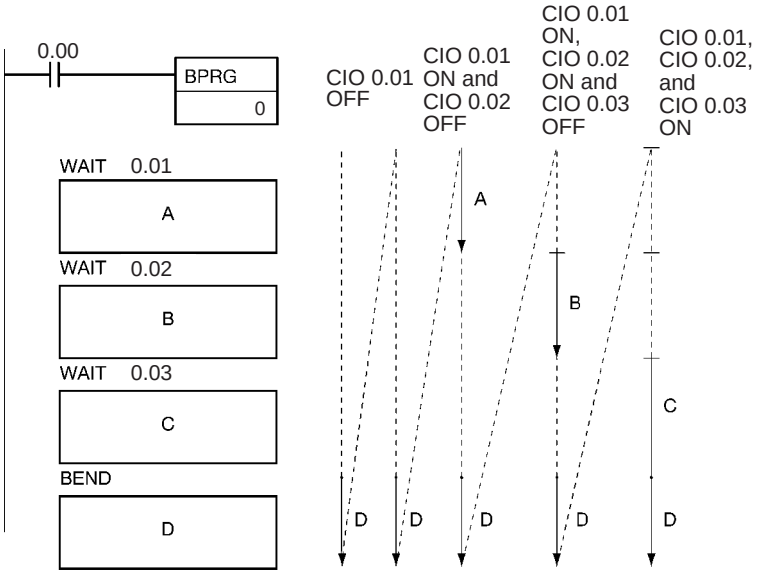
Examples

When CIO 0.00 is ON in the following example, block program 00 will be executed. Execution would proceed as follows:

1,2,3...

1. If CIO 0.01 is OFF, none of the block program will be executed until CIO 0.01 turns ON. When CIO 0.01 turns ON, “A” will be executed.
2. If CIO 0.02 is OFF after “A” is executed, the rest of the block program will not be executed until CIO 0.02 turns ON. When CIO 0.02 turns ON, “B” will be executed

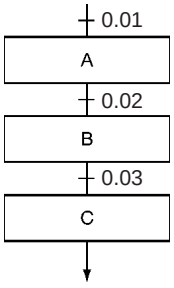
3. If CIO 0.03 is OFF after "B" is executed, the rest of the block program will not be executed until CIO 0.03 turns ON. When CIO 0.03 turns ON, "C" will be executed and the execution process will be repeated.



The following table shown the relationship between the operand bits and block program execution.

Operand bits			Program execution		
CIO 0.01	**CIO 0.02**	**CIO 0.03**	**First cycle CIO 0.00 is ON**	**Next cycle**	**Following cycles**
OFF	Any status	Any status	Nothing executed.	Nothing executed; waiting for CIO 0.01.	When CIO 0.01 turns ON "A" is executed and the status of CIO 0.02 is checked.
ON	OFF	Any status	"A" executed.	Waiting for CIO 0.02.	When CIO 0.02 turns ON "B" is executed and the status of CIO 0.03 is checked.
ON	ON	OFF	"A" and "B" executed.	Waiting for CIO 0.03.	When CIO 0.03 turns ON "C" is executed
ON	ON	ON	"A," "B," and "C" executed.	"A," "B," and "C" executed.	

As shown in this example, WAIT(805) and WAIT(805) NOT can be used to progressively execute steps within a block program.



3-30-7 TIMER WAIT: TIMW(813) and TIMWX(816)

Purpose

Delays execution of the rest of the block program until the specified time has elapsed. Execution will be continued from the next instruction after TIMW(813)/TIMWX(816) when the timer times out.

Ladder Symbol

PV Refresh Method: BCD

TIMW(813)
N
SV

N: Timer number
SV: Set value

PV Refresh Method: Binary

TIMWX(816)
N
SV

N: Timer number
SV: Set value

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

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

Note TIMW(813)/TIMWX(816) must be used in block programming regions even within subroutines.

Operands

N: Timer Number

BCD: 0 to 4095 (decimal)
Binary: 0 to 4095 (decimal)

S: Set Value

BCD: #0000 to #9999 (BCD)
Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

Operand Specifications

Area	N	SV
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A447 A448 to A959
Timer Area	0000 to 4095	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15

Area	N	SV
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	

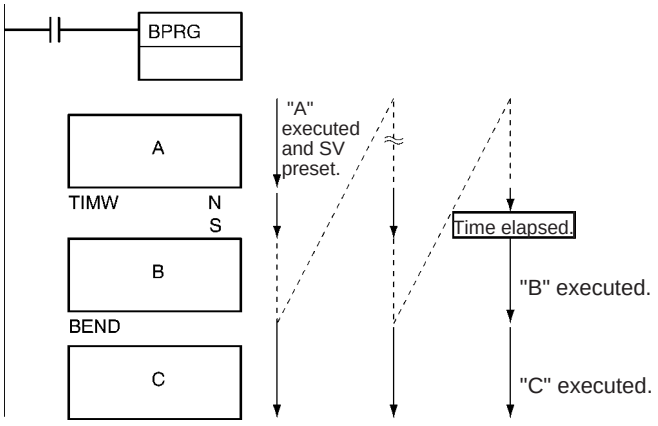
Description

TIMW(813)/TIMWX(816) creates an ON-delay countdown timer (100-ms timer set in SV) between execution of the block program instruction preceding it and the instructions following. TIMW(813) can time from 0 to 999.9 s with a timer accuracy of 0 to 0.01 s. TIMWX(816) can time from 0 to 6,553.5 s with a timer accuracy of 0 to 0.01 s.

The first part of the block program is executed the first time the block program is entered. When TIMW(813)/TIMWX(816) is reached, the Completion Flag is reset to OFF, the timer is preset to the SV, and execution of the rest of the block program will wait until SV has expired.

While the timer is timing down, only TIMW(813)/TIMWX(816) will be executed to update the timer. When the timer times out, the Completion Flag will turn ON and the rest of the block program will be executed. Once the entire block program has been executed, the process will be repeated.

TIMW(813)/TIMWX(816) can be thought of as a WAIT instruction with a timer for the execution condition and it can thus be used for timed step progressions.



Flags

Name	Label	Operation
Error Flag	ER	ON if TIMW(813)/TIMWX(816) is not in a block program. ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value. ON if in BCD mode and SV is not BCD. OFF in all other cases.

Precautions

The rest of the block program following timer will be executed if the Completion Flag for the timer is force set.

If the Completion Flag for the timer is force reset, only TIMW(813/ TIMWX(816)) will be executed in the block program until the force reset status is cleared.

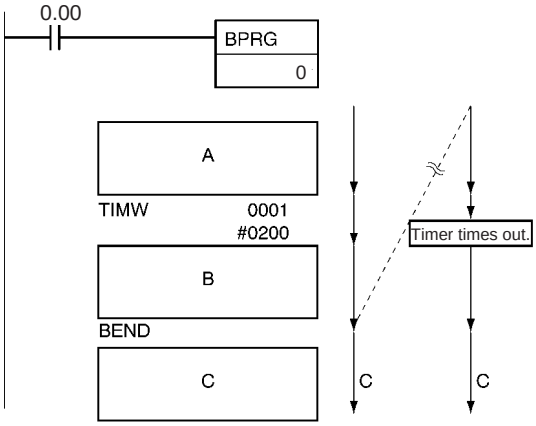
The present value of timers programmed with timer numbers T0 to T2047 will be updated even when the timer is on standby. The present value of timers programmed with timer numbers T2048 to T4095 will be held when the timer is on standby.

The timer numbers are also used by the other timer instructions. Operation will not be predictable if the same timer number is used for more than one timer instruction. Use each timer number only once. The only way that the same timer number can be used dependably is if only one of the timers is ever operating at the same time. An error will occur in the program check if the same timer number is used in more than one timer instruction.

An error will occur and the Error Flag will turn ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value or if SV is not BCD.

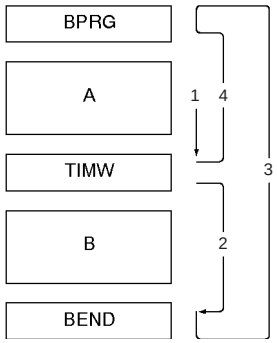
Examples

In the following example, “B” will be executed 20 seconds after “A” whenever CIO 0.00 is ON.



Address	Instruction	Operand
000200	LD	0.00
000201	BPRG	0
. .	A	. .
000210	TIMW	1
		#0200
. .	B	. .
000220	BEND	---

Program execution will flow from 2 to 3 to 4 and back to 2 during the 20 s before “B” is executed, as shown in the following diagram.



3-30-8 COUNTER WAIT: CNTW(814) and CNTWX(818)

Purpose

Delays execution of the rest of the block program until the specified count has been achieved. Execution will be continued from the next instruction after CNTW(814)/CNTWX(818) when the counter counts out.

Ladder Symbol

PV Refresh Method: BCD

CNTW(814)	N	**N**: Counter number
	SV	**SV**: Set value
	I	**I**: Count input

PV Refresh Method: Binary

CNTWX(818)	N	**N**: Counter number
	SV	**SV**: Set value
	I	**I**: Count input

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

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

Note CNTW(814)/CNTWX(818) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

N: Counter Number

BCD: 0 to 4095 (decimal)
Binary: 0 to 4095 (decimal)

S: Set Value

BCD: #0000 to #9999 (BCD)
Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

Operand Specifications

Area	N	SV	I
CIO Area	---	CIO 0 to CIO 6143	CIO 0.00 to CIO 6143.15
Work Area	---	W0 to W511	W0.00 to W511.15
Holding Bit Area	---	H0 to H511	H0.00 to H511.15
Auxiliary Bit Area	---	A0 to A447 A448 to A959	A0.00 to A447.15 A448.00 to A959.15
Timer Area	---	T0000 to T4095	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4095	C0000 to C4095
Task Flags	---		TK00 to TK31
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	---	D0 to D32767	---

Area	N	SV	I
Indirect DM addresses in binary	---	@ D0 to @ D3276	---
Indirect DM addresses in BCD	---	*D0 to *D32767	---
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)	---
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		

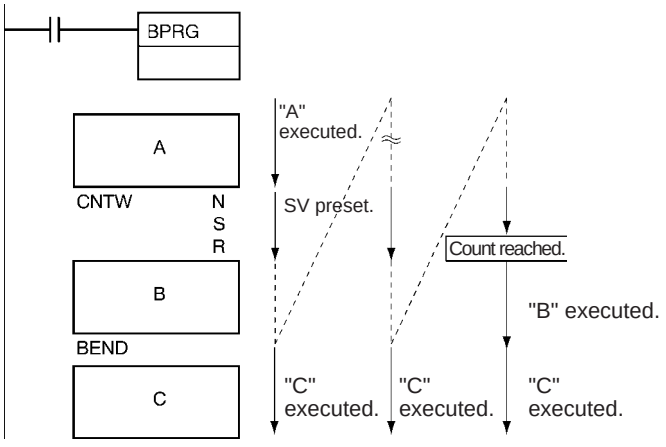
Description

CNTW(814)/CNTWX(818) creates a decrementing counter that delays execution of the instructions following it in the block program until the counter has counted out. The set value for CNTW(814) is specified in BCD between 0000 and 9999. The set value for CNTWX(818) is specified in binary between 0000 and FFFF hex.

The first part of the block program is executed the first time the block program is entered. When CNTW(814)/CNTWX(818) is reached, the Completion Flag is reset to 0, the counter is preset to SV, and execution of the rest of the block program will wait until the counter has counted out. The counter counts pulses (upward differentiation) on I, the counter input.

While the counter is counting down, only CNTW(814)/CNTWX(818) will be executed to update the counter. When the counter counts out, the Completion Flag will turn ON and the rest of the block program will be executed. Once the entire block program has been executed, the process will be repeated.

CNTW(814)/CNTWX(818) can be thought of as a WAIT instruction with a counter for the execution condition and it can thus be used for timed step progressions.



Flags

Name	Label	Operation
Error Flag	ER	ON if CNTW(814)/CNTWX(818) is not in a block program. ON if an indirect IR designation is used for N in BCD mode and the address is not for a counter present value. ON if SV is not BCD when BCD mode is set. OFF in all other cases.

Precautions

The rest of the block program following CNTW(814)/CNTWX(818) will be executed if the Completion Flag for the counter is force set.

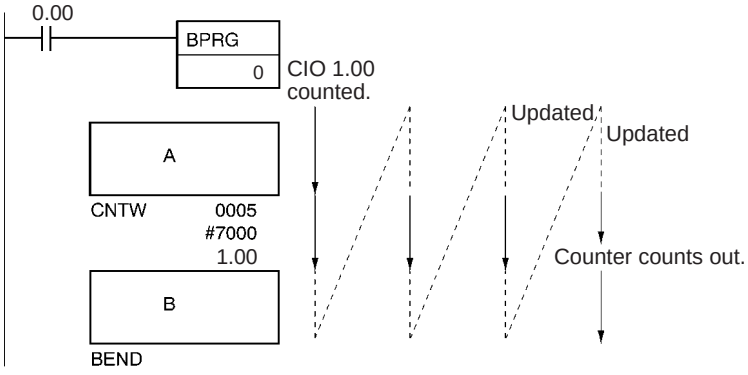
If the Completion Flag for the counter is force reset, the only CNTW(814)/CNTWX(818) will be executed in the block program until the force reset status is cleared.

The counter numbers are also used by the other counter instructions. Operation will not be predictable if the same counter number is used for more than one counter instruction. Use each counter number only once. The only way that the same counter number can be used dependably is if only one of the counters is ever operating at the same time. An error will occur in the program check if the same counter number is used in more than one counter instruction.

An error will occur and the Error Flag will turn ON if an indirect IR designation is used for N in BCD mode and the address is not for a counter present value or if SV is not BCD when BCD mode is set.

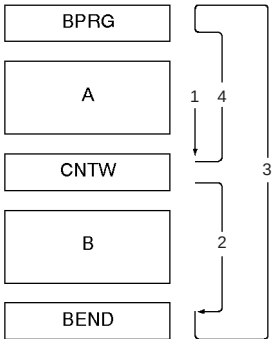
Examples

When CIO 0.00 is ON in the following example, “A” will be executed and then execution of the rest of the block program “B” will wait until 7,000 counts of CIO 1.00.



Address	Instruction	Operand
000200	LD	0.00
000201	BPRG	0
. .	A	. .
000210	CNTW	5
		#7000
		1.00
. .	B	. .
000220	BEND	---

Program execution will flow from 2 to 3 to 4 and back to 2 during the 7,000 counts before "B" is executed, as shown in the following diagram.



3-30-9 HIGH-SPEED TIMER WAIT: TMHW(815) and TMHWX(817)

Purpose

Delays execution of the rest of the block program until the specified time has elapsed. Execution will be continued from the next instruction after TMHW(815)/TMHWX(817) when the timer times out.

Ladder Symbol

PV Refresh Method: BCD

TMHW(815)	N	**N**: Timer number
	SV	**SV**: Set value

PV Refresh Method: Binary

TMHWX(817)	N	**N**: Timer number
	SV	**SV**: Set value

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

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

Note TMHW(815)/TMHWX(817) must be used in block programming regions even within subroutines.

Operands

N: Timer Number

BCD: 0 to 4095 (decimal)
Binary: 0 to 4095 (decimal)

S: Set Value

BCD: #0000 to #9999 (BCD)
Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

Operand Specifications

Area	N	SV
CIO Area	---	CIO 0 to CIO 6143
Work Area	---	W0 to W511
Holding Bit Area	---	H0 to H511
Auxiliary Bit Area	---	A0 to A447 A448 to A959
Timer Area	0000 to 4095	T0000 to T4095

Area	N	SV
Counter Area	---	C0000 to C4095
DM Area	---	D0 to D32767
Indirect DM addresses in binary	---	@ D0 to @ D32767
Indirect DM addresses in BCD	---	*D0 to *D32767
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
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	

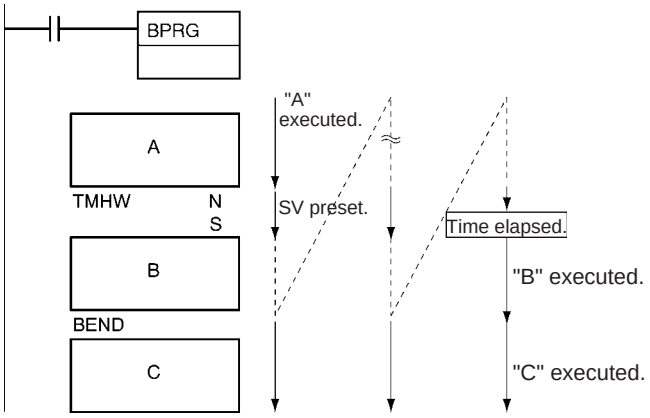
Description

TMHW(815)/TMHWX(817) creates an ON-delay countdown timer (10-ms timer set in SV) between execution of the block program instruction preceding it and the instructions following. TMHW(815) can time from 0 to 99.99 s with a timer accuracy of 0 to 0.01 s. TMHWX(817) can time from 0 to 655.35 s with a timer accuracy of 0 to 0.01 s.

The first part of the block program is executed the first time the block program is entered. When TMHW(815)/TMHWX(817) is reached, the Completion Flag is reset to OFF, the timer is preset to the SV, and execution of the rest of the block program will wait until SV has expired.

While the timer is timing down, only TMHW(815)/TMHWX(817) will be executed to update the timer. When the timer times out, the Completion Flag will turn ON and the rest of the block program will be executed. Once the entire block program has been executed, the process will be repeated.

TMHW(815)/TMHWX(817) can be thought of as a WAIT instruction with a timer for the execution condition and it can thus be used for timed step progressions.



Flags

Name	Label	Operation
Error Flag	ER	ON if TMHW(815)/TMHWX(817) is not in a block program. ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value. ON if in BCD mode and SV is not BCD. OFF in all other cases.

Precautions

The rest of the block program following TMHW(815)/TMHWX(817) will be executed if the Completion Flag for the timer is force set.

If the Completion Flag for the timer is force reset, the only TMHW(815)/TMHWX(817) will be executed in the block program until the force reset status is cleared.

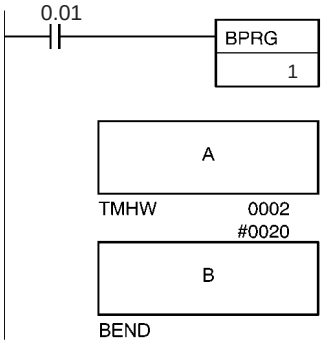
The present value of timers programmed with timer numbers T0 to T2047 will be updated even when the timer is on standby. The present value of timers programmed with timer numbers T2048 to T4095 will be held when the timer is on standby.

The timer numbers are also used by the other timer instructions. Operation will not be predictable if the same timer number is used for more than one timer instruction. Use each timer number only once. The only way that the same timer number can be used dependably is if only one of the timers is ever operating at the same time. An error will occur in the program check if the same timer number is used in more than one timer instruction.

An error will occur and the Error Flag will turn ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value or if SV is not BCD.

Examples

In the following example, “B” will be executed 20 seconds after “A” whenever CIO 0.01 is ON.



Address	Instruction	Operand
000221	LD	0.01
000222	BPRG	1
. .	A	. .
000250	TMHW	2
		#0020
. .	B	. .
000281	BEND	---

3-30-10 Loop Control: LOOP(809)/LEND(810)/LEND(810) NOT

Purpose

Create a loop that is repeatedly executed until an execution condition turns ON or OFF or until an execution condition turns ON.

Ladder Symbol

LOOP(809)

LEND(810)

LEND(810) B **B**: Bit operand

LEND(810) NOT B

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

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

Note LOOP(809), LEND(810), and LEND(810) NOT must be used in block programming regions even within subroutines and interrupt tasks.

Operand Specifications

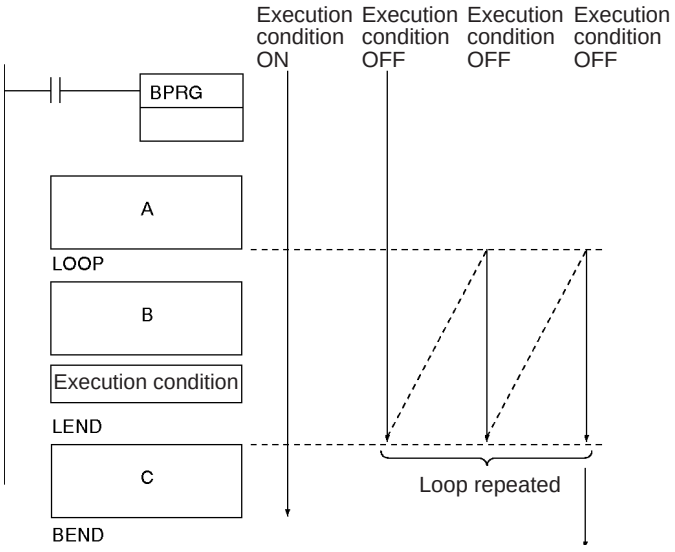
Area	B
CIO Area	CIO 0.00 to CIO 6143.15
Work Area	W0.00 to W511.15
Holding Bit Area	H0.00 to H511.15
Auxiliary Bit Area	A0.00 to A447.15 A448.00 to A959.15
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK00 to TK31
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON,OFF, AER
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 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

LOOP(809) designates the beginning of the loop program. LEND(810) or LEND(810) NOT specifies the end of the loop. When LEND(810) or LEND(810) NOT is reached, program execution will loop back to the next previous LOOP(809) until the operand bit for LEND(810) or LEND(810) NOT turns ON or OFF (respectively) or until the execution condition for LEND(810) turns ON.

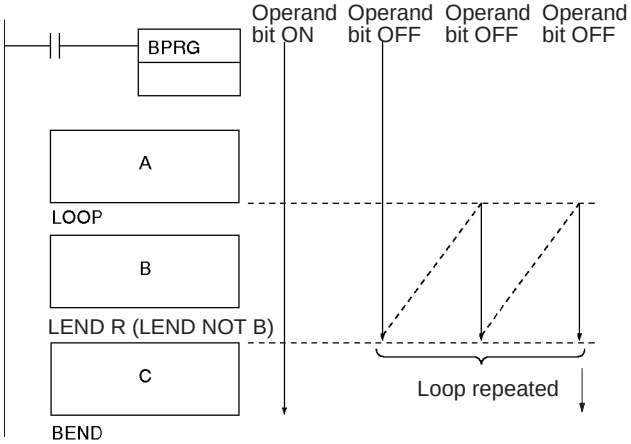
Using an Execution Condition for LEND(810)

LEND(810) can be programmed either with or without an operand bit. If an operand bit is not specified, an execution must be created before LEND(810) starting with LD. If the execution condition is OFF, execution of the loop is repeated starting with the next instruction after LOOP(809). If the execution condition is ON, the loop is ended and execution continues to the next instruction after LEND(810).



Using a Bit Operand for LEND(810) or LEND(810) NOT

Both LEND(810) and LEND(810) NOT can be programmed with an operand bit. If the operand bit is OFF for LEND(810) (or ON for LEND(810) NOT), execution of the loop is repeated starting with the next instruction after LOOP(809). If the operand bit is ON for LEND(810) (or OFF for LEND(810) NOT), the loop is ended and execution continues to the next instruction after LEND(810) or LEND(810) NOT.



Note The status of the operand bit would be reversed for LEND(810) NOT.

Note

(1) Execution inside a loop does not refresh I/O data. If I/O data must be refreshed during the loop, use IORF(184).

(2) The maximum cycle time can be exceeded if loops are repeated too long. Design the program so that the maximum cycle time is not exceeded.

Flags

Name	Label	Operation
Error Flag	ER	ON if a Loop Control Instruction is not in a block program. OFF in all other cases.

Precautions

Loops cannot be nested within loops.

Incorrect:

```
LOOP(809)
LOOP(809)
LEND(810)
LEND(810)
```

Do not reverse the order of LOOP and LEND.

Incorrect:

```
LEND(810)
   :
   :
LOOP(809)
```

Conditional block branching can be used within a loop, but the entire branch operation must be within the loop.

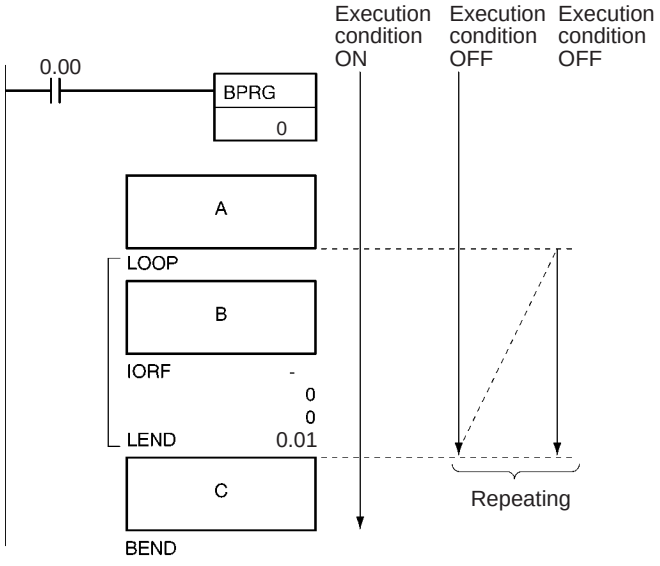
Correct:	**Incorrect:**
LOOP(809)	LOOP(809)
IF(802)	IF(802)
IF(802)	IF(802)
IEND(804)	IEND(804)
IEND(804)	LEND(810)
LEND(810)	IEND(804)

NOP processing will be performed if LOOP(809) is not executed.

An error will occur and the Error Flag will turn ON if a Loop Control Instruction is not in a block program.

Examples

When CIO 0.00 is ON in the following example, the block program is executed. After "A" is executed, "B" and the IORF(184) after it will be executed repeatedly until CIO 0.01 is ON, at which time C will be executed and the block program will end.



Address	Instruction	Operand
000220	LD	0.00
000201	BPRG	0
. .	A	. .
000210	LOOP	---
. .	B	. .
000220	IORF	. .
		0000
		0000
000221	LEND	0.01
. .	C	. .
000220	BEND	---

3-31 Text String Processing Instructions

This section describes instructions used to manipulate text strings.

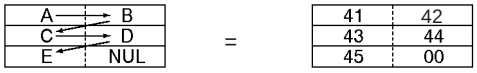
Instruction	Mnemonic	Function code	Page
MOV STRING	MOV$	664	1006
CONCATENATE STRING	+$	656	1008
GET STRING LEFT	LEFT$	652	1010
GET STRING RIGHT	RGHT$	653	1013
GET STRING MIDDLE	MID$	654	1015
FIND IN STRING	FIND$	660	1017
STRING LENGTH	LEN$	650	1019
REPLACE IN STRING	RPLC$	661	1021
DELETE STRING	DEL$	658	1023
EXCHANGE STRING	XCHG$	665	1026
CLEAR STRING	CLR$	666	1027
INSERT INTO STRING	INS$	657	1029
String Comparison Instructions	=$, <>$, <$, <=$, >$, >=$	670 to 675	1032

3-31-1 Text String Processing Overview

Data from the beginning until a NUL code (00 hex) is handled as text string data expressed in ASCII (except for 1-byte, special characters). It is stored from leftmost to rightmost bytes, and from rightmost to leftmost words.

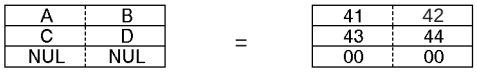
When there is an odd number of characters, 00 hex (NUL code) is stored in the available space in the rightmost byte of the final word.

Example: Text string ABCDE



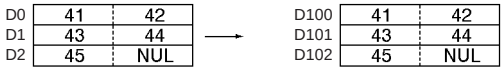
When there is an even number of characters, 0000 hex (two NUL codes) is stored in the leftmost and rightmost bytes of the word following the final word.

Example: Text string ABCD

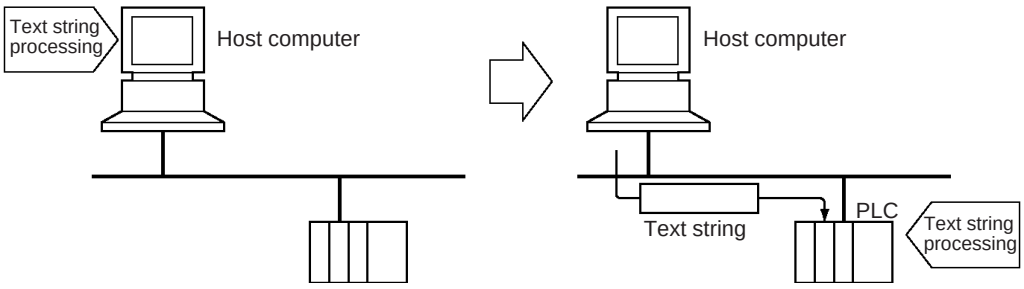


As shown in the following diagram, a text string can be specified by simply designating the first word of that string. The text string data up until the next NUL code (00 hex) will then be handled as a single block of ASCII data.

Example: MOV$ D0 D100



Text string processing instructions can be used to execute at a PLC the various kinds of text string processing (product data, and so on) that used to be executed at the host computer.



For example, production plan data such as product names can be transferred from the host computer to the PLC. Various operations such as inserting and rearranging text strings can be then be performed at the PLC, thereby reducing the data processing load at the host computer.

ASCII Characters

The ASCII characters that can be handled by text string processing instructions are shown in the following table.

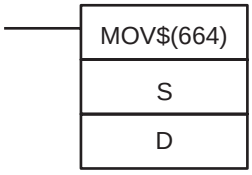
Four rightmost bits \ Four leftmost bits	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p				ー	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			”	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			’	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	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	°		

3-31-2 MOV STRING: MOV$(664)

Purpose

Transfers a text string.

Ladder Symbol



S: First source word

D: First destination word

Variations

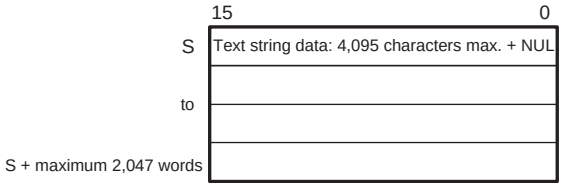
Variations		
	Executed Each Cycle for ON Condition	MOV$(664)
	Executed Once for Upward Differentiation	@MOV$(664)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

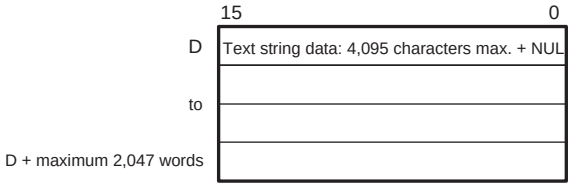
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: First Source Word



D: First Destination Word



Note

(1) The data from S to S +the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.

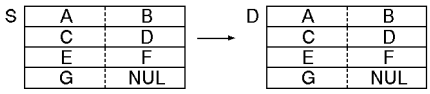
(2) The data from S to S + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A447 A448 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

MOV$(664) transfers the text string data designated by S, just as it is, as text string data (including the final NUL), to D. The maximum number of characters that can be designated by S is 4,095 (0FFF hex).



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is transferred to D. OFF in all other cases.

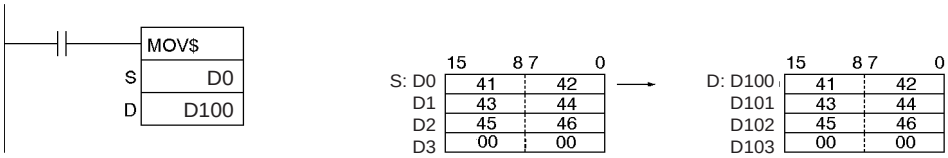
Precautions

If more than 4,095 characters are designated by S, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is transferred to D, the Equals Flag will turn ON.

Example

In this example, MOV$(664) is used to transfer the text string ABCDEF.

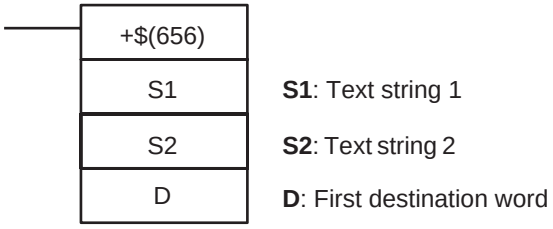


3-31-3 CONCATENATE STRING: +$(656)

Purpose

Links one text string to another text string.

Ladder Symbol



S1: Text string 1

S2: Text string 2

D: First destination word

Variations

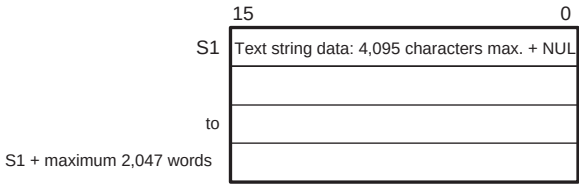
Variations		
	Executed Each Cycle for ON Condition	+$(656)
	Executed Once for Upward Differentiation	@+$(656)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

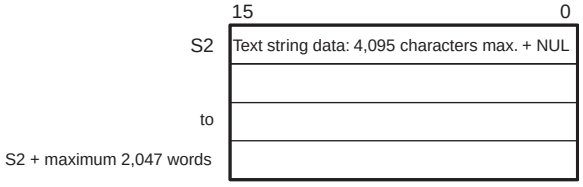
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

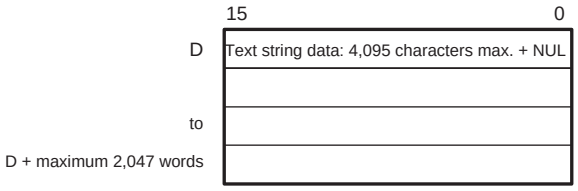
S1: Text String 1



S2: Text String 2



D: First Destination Word



Note

(1) The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from S2 to S2 + the maximum 2,047 words and from D to D + the maximum 2,047 words cannot overlap.

Operand Specifications

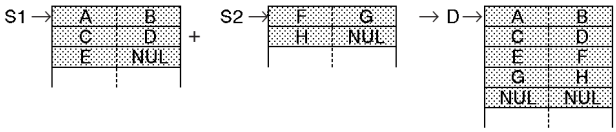
Area	S1	S2	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0V to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+$(664) connects the text string data designated by S1 to the text string data designated by S2, and outputs the result to D as text string data (including the final NUL).

The maximum number of characters that can be designated by S1 and S2 is 4,095 (0FFF hex). If there is no NUL until 4,096 characters, an error will be generated and the Error Flag will turn ON. Moreover, the result of the linkage can be no more than 4,095 characters (0FFF hex). If the linkage results in more characters than that, only the first 4,095 characters (with NUL added as the 4,096th) will be output to D.

If there is a NUL for both S1 and S2, the two NUL characters (0000 hex) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 and S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is transferred to D. OFF in all other cases.

Precautions

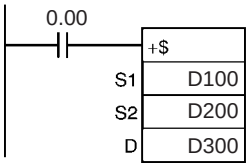
If more than 4,095 characters are designated by S1 and S2, an error will be generated and the Error Flag will turn ON.

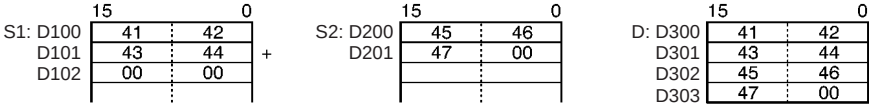
If 0000 (hex) is transferred to D, the Equals Flag will turn ON.

Do not overlap the beginning word designated by D with the character data area for S2. If they overlap, the instruction cannot be executed properly.

Example

In this example, +$(656) is used to connect the text strings ABCD and EFG and output the result to D.



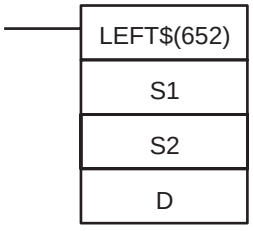


3-31-4 GET STRING LEFT: LEFT$(652)

Purpose

Fetches a designated number of characters from the left (beginning) of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

D: First destination word

Variations

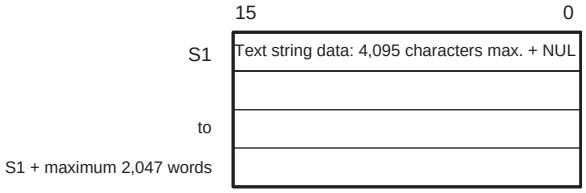
Variations		
	Executed Each Cycle for ON Condition	LEFT$(652)
	Executed Once for Upward Differentiation	@LEFT$(652)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

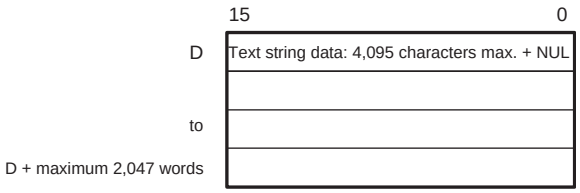
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)



Note

(1) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

Area	S1	S2	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	---
Data Registers	---	DR0 to DR15	---

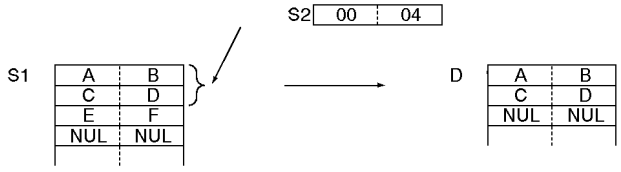
Area	S1	S2	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

LEFT$(652) reads the number of characters designated by S2, from the left (the beginning) of the first word of the text string designated by S1 until the NUL code (00 hex), and outputs the result to D (with NUL added at the end).

If the number of characters fetched exceeds the number of characters designated by S1, the entire S1 text string will be output.

If 0 (0000 hex) is designated as the number of characters to be read, the two NUL characters (0000 hex) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

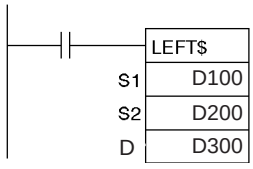
Precautions

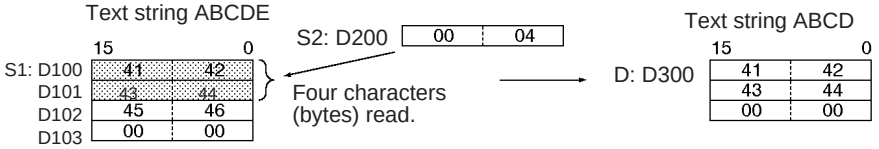
The maximum number of characters to be read that can be designated by S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, LEFT$(652) is used to read four characters.



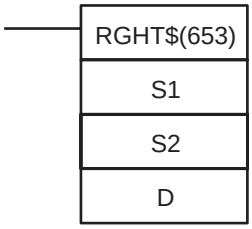


3-31-5 GET STRING RIGHT: RGHT$(653)

Purpose

Reads a designated number of characters from the right (end) of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

D: First destination word

Variations

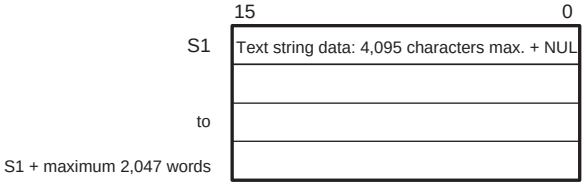
Variations		
	Executed Each Cycle for ON Condition	RGHT$(653)
	Executed Once for Upward Differentiation	@RGHT$(653)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

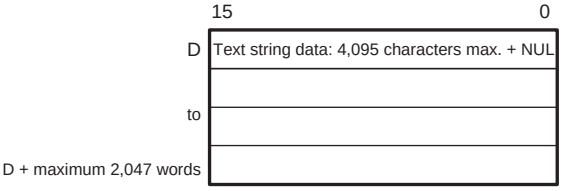
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)



Note

(1) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

Area	S1	S2	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		

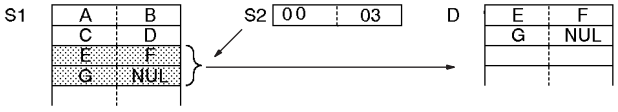
Area	S1	S2	D
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	---
Data Registers	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

RGHT$(653) reads the number of characters designated by S2, from the left (the beginning) of the first word of the text string designated by S1 until the NUL code (00 hex), and outputs the result to D (with NUL added at the end).

If the number of characters to be read exceeds the number of characters designated by S1, the entire S1 text string will be output.

If 0 (0000 hex) is designated as the number of characters to be read, the two NUL characters (0000 hex) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

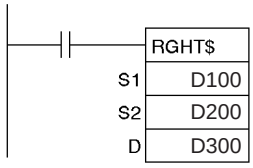
Precautions

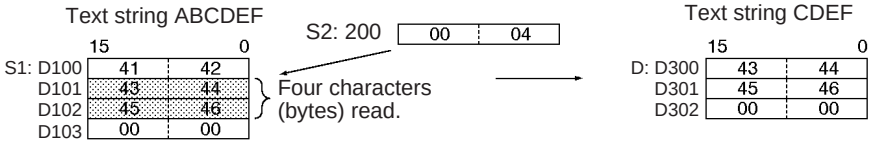
The maximum number of characters to be read that can be designated by S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, RGHT$(653) is used to read four characters.



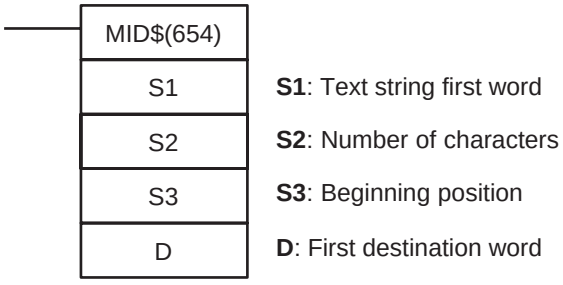


3-31-6 GET STRING MIDDLE: MID$(654)

Purpose

Reads a designated number of characters from any position in the middle of a text string.

Ladder Symbol



Variations

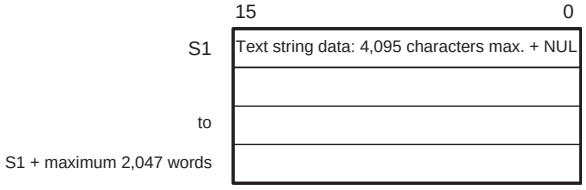
Variations		
Variations	**Executed Each Cycle for ON Condition**	MID$(654)
	Executed Once for Upward Differentiation	@MID$(654)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

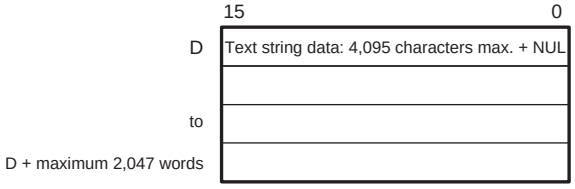
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)
S3: Beginning Position (0001 to 0FFF hex or &1 to &4095)



Note

(1) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

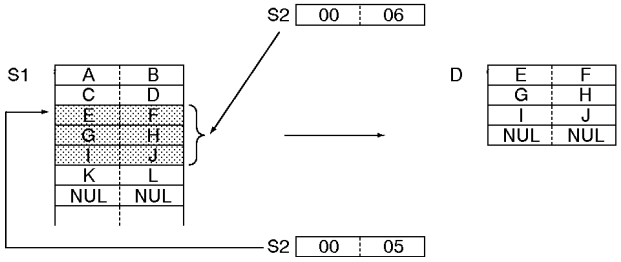
Area	S1	S2	S3	D
CIO Area	CIO 0 to CIO 6143			
Work Area	W0 to W511			
Holding Bit Area	H0 to H511			
Auxiliary Bit Area	A0 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			

Area	S1	S2	S3	D
Counter Area	C0000 to C4095			
DM Area	D0 to 32767			
Indirect DM addresses in binary	@ D0 to @ D32767			
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	#0001 to #0FFF (binary) or &1 to &4095	---
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

Within the text string identified by the first word designated by S1 until the NUL code (00 hex), MID$(654) reads the number of characters designated by S2, from the beginning word designated by S3, and outputs the result to D as text string data (with NUL added at the end).

If the number of characters to be read extends beyond the end of the text string designated by S1, the string will be output up to the end.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the S3 data is within the range of 1 to 4,095 (0001 to 0FFF hex). ON if S3 is greater than S1. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

Precautions

The range for the beginning position designated by S3 is the 1st to the 4,095th character (0001 to 0FFF hex). If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

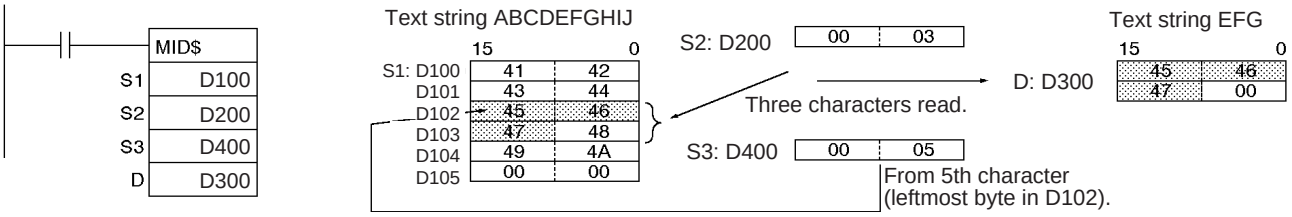
The maximum number of characters to be read that can be designated by S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0 (0000 hex) is designated as the number of characters to be read, the two NUL characters (0000 hex) will be output to D.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, MID$(654) is used to read three characters.

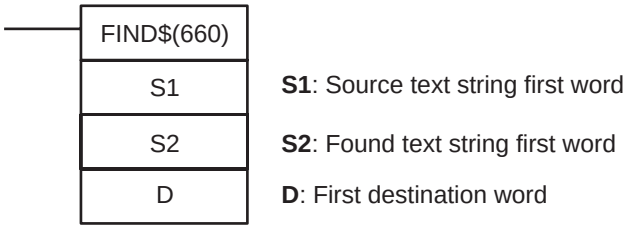


3-31-7 FIND IN STRING: FIND$(660)

Purpose

Finds a designated text string from within a text string.

Ladder Symbol



S1: Source text string first word

S2: Found text string first word

D: First destination word

Variations

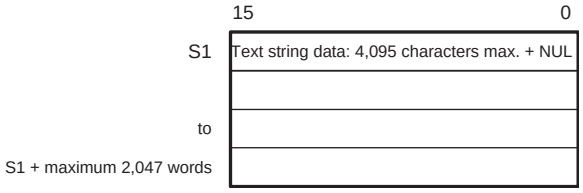
Variations		
	Executed Each Cycle for ON Condition	FIND$(660)
	Executed Once for Upward Differentiation	@FIND$(660)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

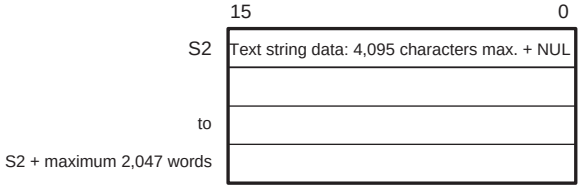
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Source Text String



S2: Found Text String



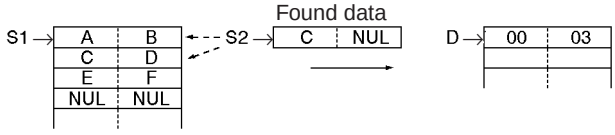
Note The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words must be in the same area.

Operand Specifications

Area	S1	S2	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

FIND$(660) finds the text string designated by S2 from within the text string designated by S1, and outputs the result (a given number of characters from the beginning of S1) in binary data to D. If there is no matching text string, 0000 hex is output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

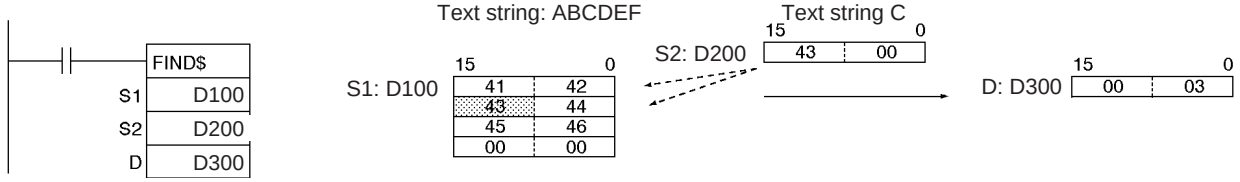
Precautions

The maximum number of characters to be read that can be designated by S1 or S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, FIND$(660) is used to find one character from within a text string.

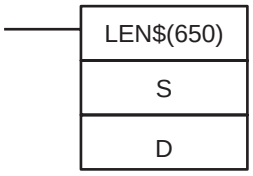


3-31-8 STRING LENGTH: LEN$(650)

Purpose

Calculates the length of a text string.

Ladder Symbol



S: Text string first word

D*:* First destination word

Variations

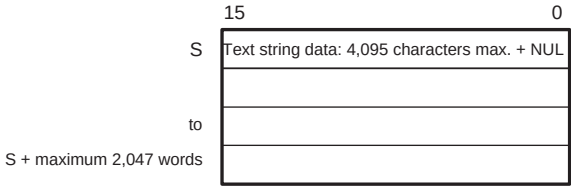
Variations		
	Executed Each Cycle for ON Condition	LEN$(650)
	Executed Once for Upward Differentiation	@LEN$(650)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: Text String



Note The data from S to S + the maximum 2,047 words must be in the same area.

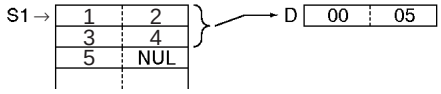
Operand Specifications

Area	S	D
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A447 A448 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	
Constants	---	

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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

LENS$(650) calculates the number of characters from the first word of the text string, designated by S, until the NUL code (00 hex), including the NUL code itself, and outputs the result to D as binary data. If there is a NUL at the beginning of the text string, the result that is calculated will be 0000 hex.



Flags

Name	Label	Operation
Error Flag	ER	ON if the calculated result comes to more than 4,095 characters. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if the calculated result is 0. OFF in all other cases.

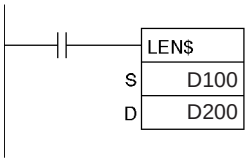
Precautions

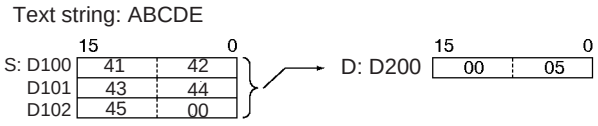
The maximum number of characters is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, LENS$(650) is used to calculate the number of characters and output the result.



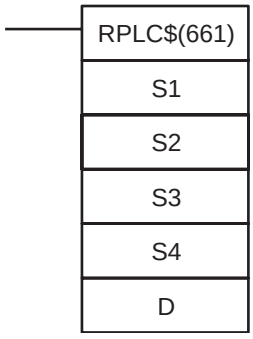


3-31-9 REPLACE IN STRING: RPLC$(661)

Purpose

Replaces a text string with a designated text string from a designated position.

Ladder Symbol



S1: Text string first word

S2: Replacement text string first word

S3: Number of characters

S4: Beginning position

D: First destination word

Variations

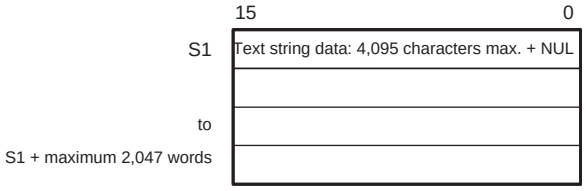
Variations		
	Executed Each Cycle for ON Condition	RPLC$(661)
	Executed Once for Upward Differentiation	@RPLC$(661)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

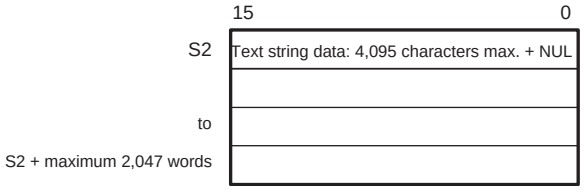
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

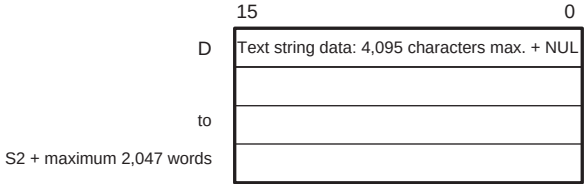
S1: Text String



S2: Replacement Text String



S3: Number of Characters (0000 to 0FFF hex or &0 to &4095)
S4: Beginning Position (0001 to 0FFF hex or &0 to &4095)



Note

(1) The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from D to D + the maximum 2,047 words and from either S1 to S1 + the maximum 2,047 words or from S2 to S2 + the maximum 2,047 words can overlap.

Operand Specifications

Area	S1	S2	S3	S4	D
CIO Area	CIO 0 to CIO 6143				
Work Area	W0 to W511				
Holding Bit Area	H0 to H511				
Auxiliary Bit Area	A0 to A447 A448 to A959				A448 to A959
Timer Area	T0000 to T4095				
Counter Area	C0000 to C4095				
DM Area	D0 to D32767				
Indirect DM addresses in binary	@ D0 to @ D32767				
Indirect DM addresses in BCD	*D0 to *D32767				
Constants	---		#0000 to #0FFF (binary) or &0 to &4095	#0001 to #0FFF (binary) or &1 to &4095	---
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				

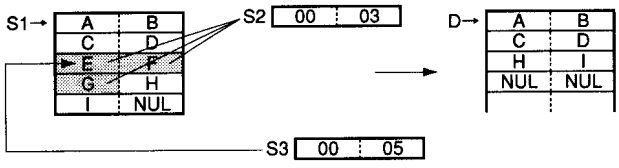
Description

RPLC$(661) replaces part of the text string designated by S1, from the beginning position designated by S4, with the text string designated by S2, and outputs the result to D as text string data (with NUL added at the end). The number of characters to be replaced is designated by S3.

The maximum number of characters in the result is 4,095 (0FFF hex). If the number is greater than that, only 4,095 characters will be output (with NUL added as the 4,096th).

From 0 to 4,095 characters (0000 to 0FFF hex) can be replaced. If the number is 0, then the text string designated by S1 will be output to D just as it is, with no change. If the S2 text string is NUL, then the operation will be the same as deleting the designated range of text in S1.

If the S1 text string from beginning to end is replaced by NUL, then two NUL characters (0000 hex) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. ON if more than 4,095 characters (0FFF hex) are designated by S3. ON if the S4 data is within the range of 1 to 4,095 (0001 to 0FFF hex). ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

Precautions

The maximum number of characters for S1 or S2 is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

The range for the beginning position designated by S4 is the 1st to the 4,095th character (0001 to 0FFF hex). If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

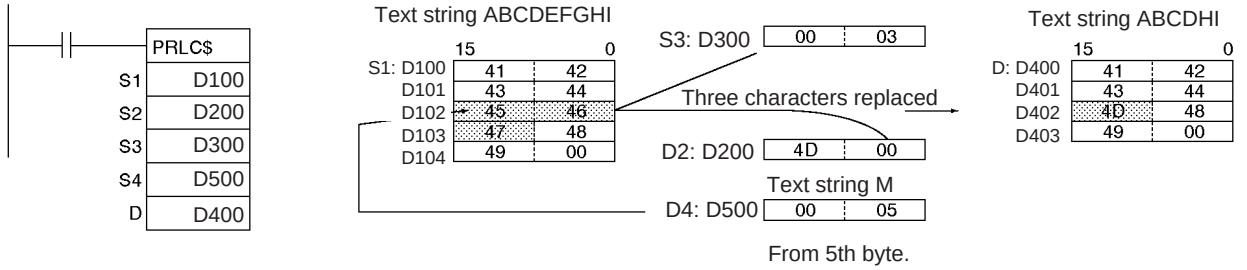
If the beginning position designated by S4 is beyond the text string designated by S1, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Set the first destination word D so that it does not overlap with the areas set with the replacement text string first word S2. RPLC$(654) will not work correctly if these areas overlap.

Example

In this example, RPLC$(654) is used to read three characters.

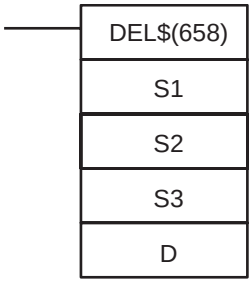


3-31-10 DELETE STRING: DEL$(658)

Purpose

Deletes a designated text string from the middle of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

S3: Beginning position

D: First destination word

Variations

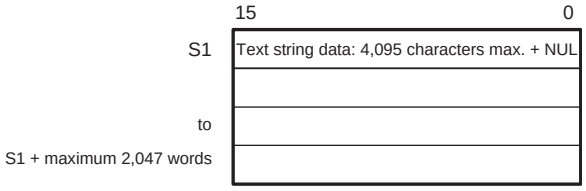
Variations		
	Executed Each Cycle for ON Condition	DEL$(658)
	Executed Once for Upward Differentiation	@DEL$(658)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

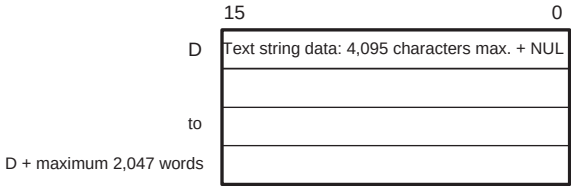
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)
S3: Beginning Position (0001 to 0FFF hex or &1 to &4095)



Note

(1) The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

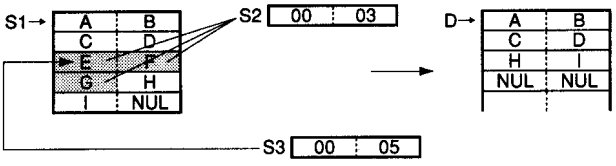
Operand Specifications

Area	S1	S2	S3	D
CIO Area	CIO 0 to CIO 6143			
Work Area	W0 to W511			
Holding Bit Area	H0 to H511			
Auxiliary Bit Area	A0 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D0 to D32767			
Indirect DM addresses in binary	@ D0 to @ D32767			
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	#0001 to #0FFF (binary) or &1 to &4095	---
Data Registers	---	DR0 to DR15		---

Area	S1	S2	S3	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

Within the text string designated by S1, DEL$(658) deletes the number of characters designated by S2, from the beginning word designated by S3, and outputs the result to D as text string data (with NUL added at the end).



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the S3 data is within the range of 1 to 4,095 (0001 to 0FFF hex). ON if S3 is greater than S1. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON when 0000 hex is output to D. OFF in all other cases.

Precautions

The maximum number of characters for S1 is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

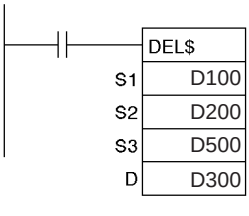
The range for the beginning position designated by S3 is the 1st to the 4,095th character (0001 to 0FFF hex). If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

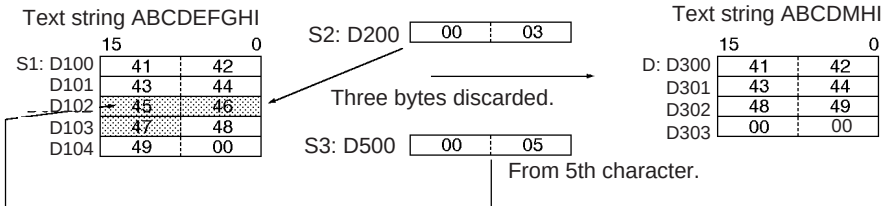
If the number of words specified for S1 exceeds the length of the text string, the Error Flag will turn ON.

If the number of characters to be deleted extends beyond the end of the S1 text string, all of the characters up to the end will be deleted. If all of the characters from the beginning of S1 to the end are designated to be deleted, then 000 hex will be output to D.

Example

In this example, DEL$(658) is used to read three characters.



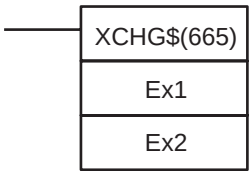


3-31-11 EXCHANGE STRING: XCHG$(665)

Purpose

Replaces a designated text string with another designated text string.

Ladder Symbol



Ex1: First exchange word 1

Ex2: First exchange word 2

Variations

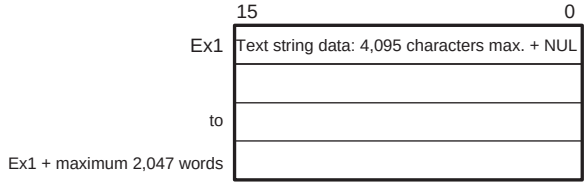
Variations		
	Executed Each Cycle for ON Condition	XCHG$(665)
	Executed Once for Upward Differentiation	@XCHG$(665)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

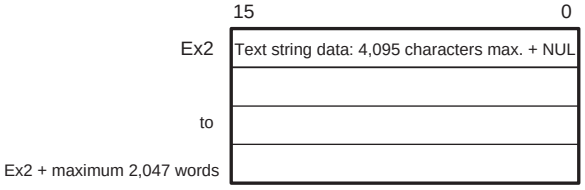
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

Ex1: First Exchange Word 1



Ex2: First Exchange Word 2



Note

(1) The data from Ex1 to Ex1 + the maximum 2,047 words and from Ex2 to Ex2 + the maximum 2,047 words must be in the same area.

(2) The data from Ex1 to Ex1 + the maximum 2,047 words and from Ex2 to Ex2 + the maximum 2,047 words cannot overlap.

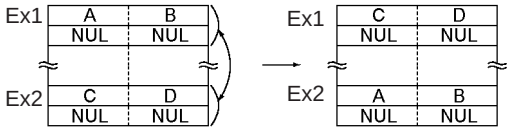
Operand Specifications

Area	Ex1	Ex2
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 to *D32767	

Area	Ex1	Ex2
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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

XCHG$(665) exchanges the text string designated by Ex1 with the text string designated by Ex2. If either Ex1 or Ex2 is NUL, then two NUL characters (0000 hex) will be output to the other one of them.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by Ex1 or Ex2. ON the Ex1 and Ex2 data overlap. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

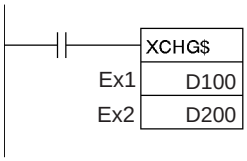
Precautions

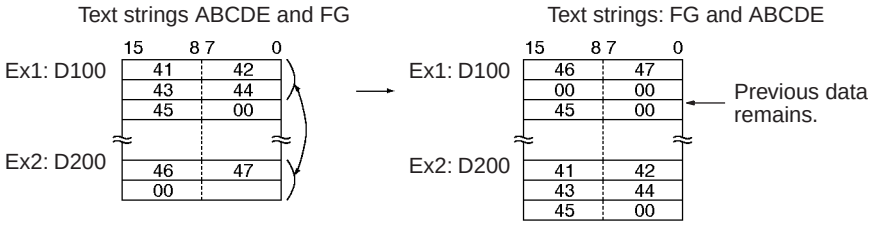
The maximum number of characters that can be designated by Ex1 or Ex2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If the text string data designated by Ex1 and Ex2 overlaps, an error will be generated and the Error Flag will turn ON.

Example

In this example, XCHG$(665) is used to exchange two text strings.



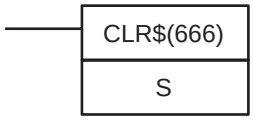


3-31-12 CLEAR STRING: CLR$(666)

Purpose

Clears an entire text string with NUL (00 hex).

Ladder Symbol



S: Text string first word

Variations

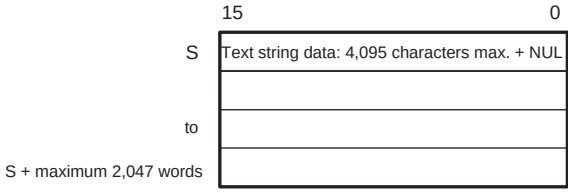
Variations		
	Executed Each Cycle for ON Condition	CLR$(666)
	Executed Once for Upward Differentiation	@CLR$(666)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: Text String First Word



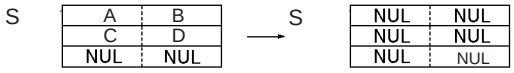
Note The data from S to S + the maximum 2,047 words must be in the same area.

Operand Specifications

Area	S
CIO Area	CIO 0 to CIO 6143
Work Area	W0 to W511
Holding Bit Area	H0 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D0 to D32767
Indirect DM addresses in binary	@ D0 to @ D32767
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

CLR$(666) clears with NUL (00 hex) the entire text string from the first word designated by S until the NUL code (00 hex). The maximum number of characters that can be cleared is 4,096. If there is no NUL before the 4,096 character, only 4,096 characters will be cleared.

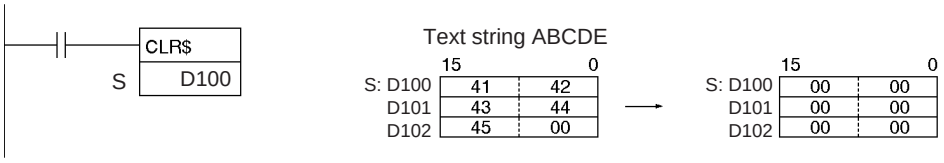


Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Example

In this example, CLR$(666) is used to clear text string ABCDE.

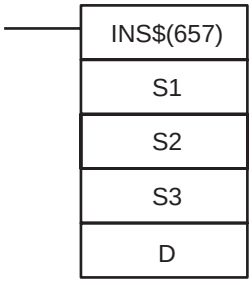


3-31-13 INSERT INTO STRING: INS$(657)

Purpose

Deletes a designated text string from the middle of a text string.

Ladder Symbol



S1: Base text string first word

S2: Inserted text string first word

S3: Beginning position

D: First destination word

Variations

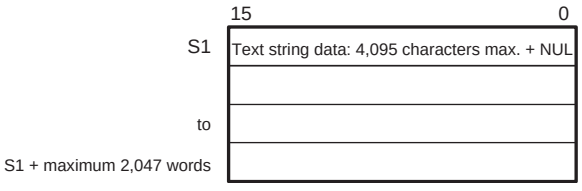
Variations		
	Executed Each Cycle for ON Condition	INS$(657)
	Executed Once for Upward Differentiation	@INS$(657)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

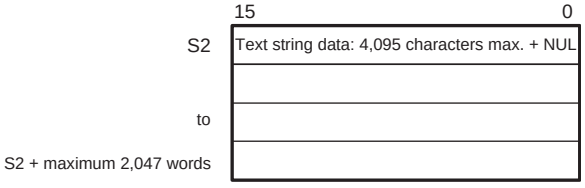
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

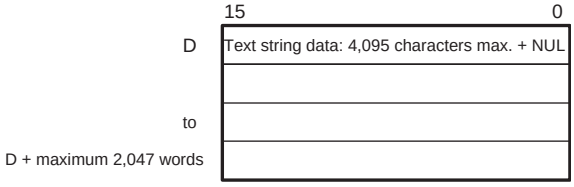
S1: Base Text String



S2: Inserted Text String



S3: Beginning Position (0000 to 0FFF hex or &0 to &4095)



Note

(1) The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.

(2) The data from S2 to S2 + the maximum 2,047 words and from D to D + the maximum 2,047 words cannot overlap. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap. The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words can also overlap.

Operand Specifications

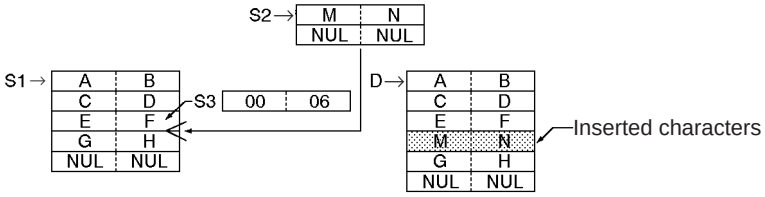
Area	S1	S2	S3	D
CIO Area	CIO 0 to CIO 6143			
Work Area	W0 to W511			
Holding Bit Area	H0 to H511			
Auxiliary Bit Area	A0 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D0 to D32767			
Indirect DM addresses in binary	@ D0 to @ D32767			
Indirect DM addresses in BCD	*D0 to *D32767			
Constants	---		#0000 to #0FFF (binary) or &0 to &4095	---
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

Within the text string designated by S1, INS$(657) inserts the text string designated by S2, after the beginning word designated by S3, and outputs the result to D as text string data (with NUL added at the end).

The maximum number of characters that can be inserted is 4,095 (0FFF hex). If there are more than that, only 4,095 characters will be output to D (with NUL added as the 4,096th character).

If either S1 or S2 is NUL, then the text string designated by the other one of them will be output to D just as it is. If S1 and S2 are both NUL, then two NUL characters (0000 hex) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. ON if S3 exceeds 4,095 (0FFF hex). ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

Precautions

The maximum number of characters for S1 and S2 is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

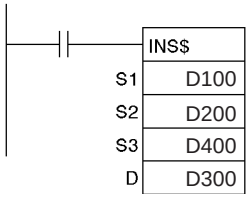
The range for the beginning position designated by S3 is 0 to 4,095. If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

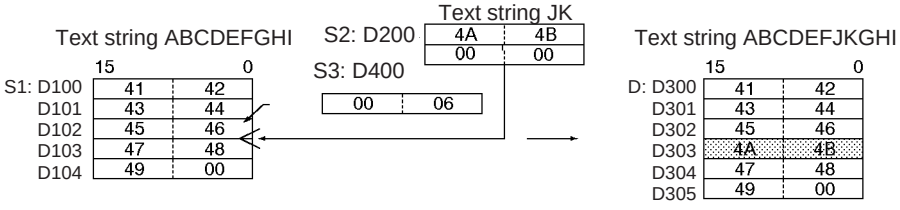
If 0000 (hex) is output to D, the Equals Flag will turn ON.

Do not overlap the destination words designated by D with the text string data designated by S2. If these overlap, the operation will not be executed properly.

Example

In this example, INS$(657) is used to insert two characters.





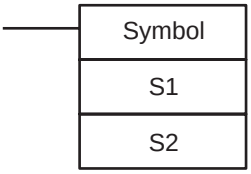
3-31-14 String Comparison Instructions (670 to 675)

Purpose

Sting comparison instructions (=$, <>$, <$, <=$, >$, >=$) compare two text strings from the beginning, in terms of value of the ASCII codes. If the result of the comparison is true, an ON execution condition is created for a LOAD, AND, or OR.

Ladder Symbol

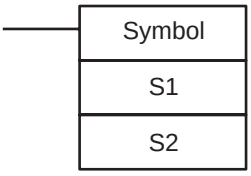
LD (Load)



S1: Text string 1

S2: Text string 2

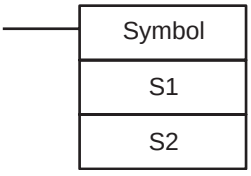
AND (Series Connection)



S1: Text string 1

S2: Text string 2

OR (Parallel Connection)



S1: Text string 1

S2: Text string 2

Variations

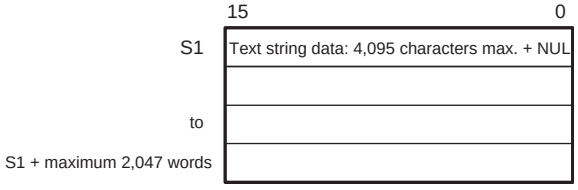
Variations	Creates ON Each Cycle Comparison is True	String comparison instructions
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

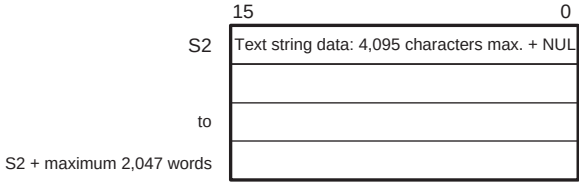
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String 1



S2: Text String 2



Note

(1) The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words be in the same area.

(2) The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words cannot overlap.

Operand Specifications

Area	S1	S2
CIO Area	CIO 0 to CIO 6143	
Work Area	W0 to W511	
Holding Bit Area	H0 to H511	
Auxiliary Bit Area	A0 to A447 A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D0 to D32767	
Indirect DM addresses in binary	@ D0 to @ D32767	
Indirect DM addresses in BCD	*D0 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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

String comparison instructions compare the text strings designated by S1 and S2. If the result of the comparison is true, an ON execution condition is created in the ladder diagram. The maximum number of characters for either S1 or S2 is 4,095 (0FFF hex).

String comparison instructions are expressed using the 18 different mnemonics listed below. (LD, AND, and OR do not appear in the ladder diagram.)

LD=$, AND=$, OR=$
LD<>$, AND<>$, OR<>$
LD<$, AND<$, OR<$
LD<=$, AND<=$, OR<=$
LD>$, AND>$, OR>$
LD>=$, AND>=$, OR>=$

The following table provides details on these instructions.

Mnemonic (including function code)	Name	Function
LD=$(670)	LOAD STRING EQUALS	True when S1 text string equals S2 text string.
AND=$(670)	AND STRING EQUALS	
OR=$(670)	OR STRING EQUALS	
LD<>$(671)	LOAD STRING NOT EQUAL	True when S1 text string does not equal S2 text string.
AND<>$(671)	AND STRING NOT EQUAL	
OR<>$(671)	OR STRING NOT EQUAL	
LD<$(672)	LOAD STRING LESS THAN	True when S1 text string is less than S2 text string.
AND<$(672)	AND STRING LESS THAN	
OR<$(672)	OR STRING LESS THAN	

Mnemonic (including function code)	Name	Function
LD<=$(673)	LOAD STRING LESS THAN OR EQUALS	True when S1 text string is less than or equal to S2 text string.
AND<=$(673)	AND STRING LESS THAN OR EQUALS	
OR<=$(673)	OR STRING LESS THAN OR EQUALS	
LD>$(674)	LOAD STRING GREATER THAN	True when S1 text string is greater than S2 text string.
AND>$(674)	AND STRING GREATER THAN	
OR>$(674)	OR STRING GREATER THAN	
LD>=$(675)	LOAD STRING GREATER THAN OR EQUALS	True when S1 text string is greater than or equal to S2 text string.
AND>=$(675)	AND STRING GREATER THAN OR EQUALS	
OR>=$(675)	OR STRING GREATER THAN OR EQUALS	

Comparison Methods

The comparison methods are as follows:

The first character (byte) of each text string is compared with its counterpart from the other string as ASCII code. If the two ASCII codes are not equal, then that greater/lesser relationship becomes the greater/lesser relationship for the two text strings. If the two ASCII codes are equal, the next characters are compared. If these two ASCII codes are not equal, then, that greater/lesser relationship becomes the greater/lesser relationship for the two text strings.

In this manner, the two text strings are compared in order, character by character. If all of the characters, including the NUL, are equal, then the two text strings will have an equal relationship.

If the two text strings are of differing lengths, then the NUL (00 hex) will be added to the shorter of the two strings to fill in the difference, and the comparison will be made on that basis.

Comparison Examples

AD (414400 hex) and BC (424300 hex):
AD < BC, because at the beginning of the text strings 41 (hex) is less than 42 (hex).

ADC (41444300 hex) and B (4200 hex):
ADC < B, because at the beginning of the text strings 41 (hex) is less than 42 (hex).

ABC (41424300 hex) and ABD (41424400 hex):
ABC < ABD, because at the beginning of the text strings the 41s and 42s match, so the result is determined by 43 being less than 44.

ABC (41424300 hex) and AB (414200 hex):
ABC > AB, because at the beginning of the text strings the 41s and 42s match, so the result is determined by 43 being greater than 00.

AB (414200 hex) and AB (414200 hex):
AB = AB, because the 41s, the 42s, and the 00s all match.

Continue programming one instruction after another, treating LD, AND, and OR in the same way. LD and OR instructions can be connected directly to the bus bar, but AND instructions cannot.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. OFF in all other cases.
Greater Than Flag	>	ON if the comparison results in S1 greater than S2. OFF in all other cases.
Greater Than or Equals Flag	>=	ON if the comparison results in S1 greater than or equal to S2. OFF in all other cases.
Equals Flag	=	ON if the comparison results in S1 equal to S2. OFF in all other cases.
Not Equal Flag	<>	ON if the comparison results in S1 not equal to S2. OFF in all other cases.
Less Than Flag	<	ON if the comparison results in S1 less than S2. OFF in all other cases.
Less Than or Equals Flag	<=	ON if the comparison results in S1 less than or equal to S2. OFF in all other cases.

Note String comparison instructions are used to rearrange the order of text strings in order of ASCII. For example, the ASCII order from lower to higher is the order of the alphabet from A to Z, so text strings can be arranged in alphabetical order.

Precautions

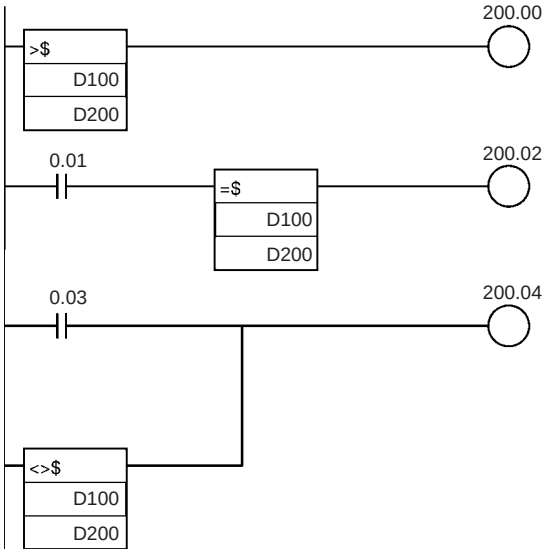
Please a right-hand instruction after these instructions. The String Comparison Instructions cannot appear on the right side of the ladder diagram.

These instructions cannot be used on the last rung of a logic block.

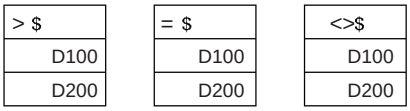
The maximum number of characters that can be compared is 4,095 (0FFF hex). If that number is exceeded (i.e., if there is no NUL before the 4,096th character), an error will occur and the Error Flag will turn ON. When this happens, an OFF execution condition will be output to the next instruction.

Example

In this example, string comparison instructions are used to compare data.



Address	Mnemonic	Operand
000000	LD > $	--- D100 D200
000001	OUT	100.00
000002	LD	0.01
000003	AND=$	--- D100 D200
000004	OUT	200.02
000005	LD	0.03
000006	OR <> $	--- D100 D200
000007	OUT	200.04



Text string ABCD

D100	41	42
D101	44	43
D102	00	00

Text string ABC

D200	41	42
D201	43	00

ON OFF ON

Text string ABC

D100	41	42
D101	43	00

Text string ABC

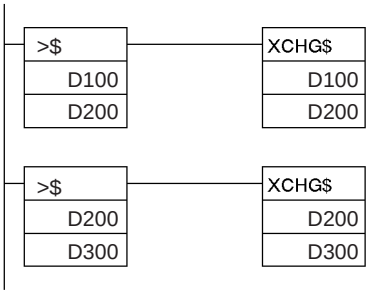
D200	41	42
D201	43	00

OFF ON OFF

In this example, three text strings are rearranged in alphabetical order. The original order is as follows:

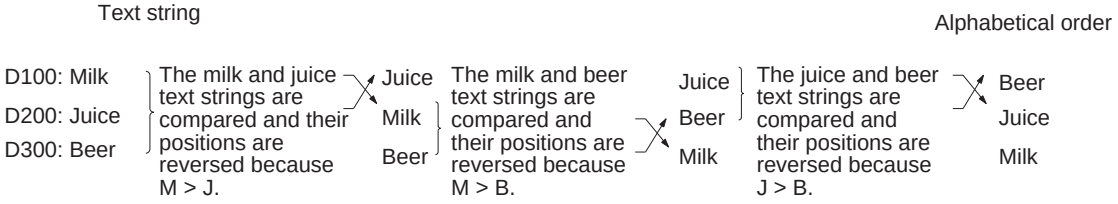
D100: Milk
D200: Juice
D300: Beer

When rearranged alphabetically, the order is as follows: beer, juice, milk.



Two text strings beginning with D100 and D200 are compared in ASCII order from lower to higher. If the text string beginning with D100 is higher in ASCII order than the one beginning with D200, then the position of the two text strings will be reversed.

Two text strings beginning with D200 and D300 are compared in ASCII order from lower to higher. If the text string beginning with D200 is higher in ASCII order than the one beginning with D300, then the position of the two text strings will be reversed.



In this way, three text strings can be rearranged in alphabetical order.

3-32 Task Control Instructions

This section describes instructions used to control tasks.

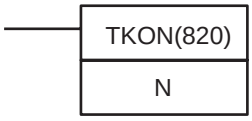
Instruction	Mnemonic	Function code	Page
TASK ON	TKON	820	1037
TASK OFF	TKOF	821	1040

3-32-1 TASK ON: TKON(820)

Purpose

Makes the specified task executable. Also, causes an interrupt task to operate as an extra cyclic task.

Ladder Symbol



N: Task number

Variations

Variations		
	Executed Each Cycle for ON Condition	TKON(820)
	Executed Once for Upward Differentiation	@TKON(820)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Task number

The allowed range for N depends on the kind of task being specified.

- Cyclic tasks:
  N must be a constant between 0 and 31 decimal. (Values 0 to 31 specify cyclic tasks 0 to 31.)
- Extra cyclic tasks:
  N must be a constant between 8000 and 8255 decimal. (Values 8000 to 8255 specify extra cyclic tasks 0 to 255.)

Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	00 to 31 or 8000 to 8255 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

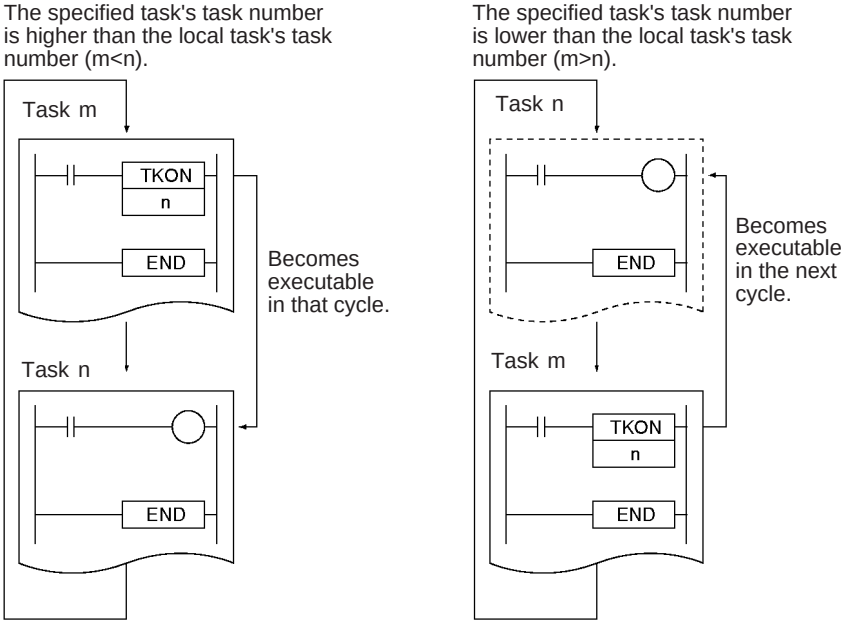
Description

TKON(820) puts the specified cyclic task or extra cyclic task in executable status. When N is 0 to 31 (specifying a cyclic task), the corresponding Task Flag (TK00 to TK31) will be turned ON at the same time.

This instruction can be executed only in a regular cyclic task or an extra cyclic task. An error will occur if an attempt is made to execute it in an interrupt task.

The cyclic task or extra cyclic task specified in TKON(820) will be also be executable in later cycles as long as it is not put in standby status by TKOF(821).

Any task can be made executable from any cyclic task, although the specified task will not be executed until the next cycle if its task number is lower than the task number of the local task. The task will be executed in the same cycle if its task number is higher than the local task's task number.



TKON(820) will be treated as NOP(000) if the specified task is already executable or the local task is specified.

A task in executable status can be put in standby status with TKOF(821), the CX-Programmer, or a FINS command.

The terms executable and executing are not interchangeable. Executable tasks are executed in order of their task numbers during cyclic program execution. An executable task will not be executed if it is put in standby status before program execution reaches its task number.

Note

(1) The CX-Programmer's *General Properties Tab* for each task has a setting (the *Operation start* box) that specifies whether the cyclic task will be executable at startup. When the *Operation start* box has been checked, the corresponding cyclic task will be put in executable status automatically when the PLC begins operation. All other cyclic tasks will be in non-executable status.

(2) If a task is in non-executable status, TKON(820) can executed to put that task into executable status. Likewise, a cyclic task in executable status can be put into non-executable status with the TKOF(821) instruction.

(3) Cyclic tasks or extra cyclic tasks that were made executable will be put in executable status in that cycle in task-number order. Consequently, a task will not be executed if it is put into standby status before the cycle's processing reaches that task as each task is executed in task-number order.

Flags

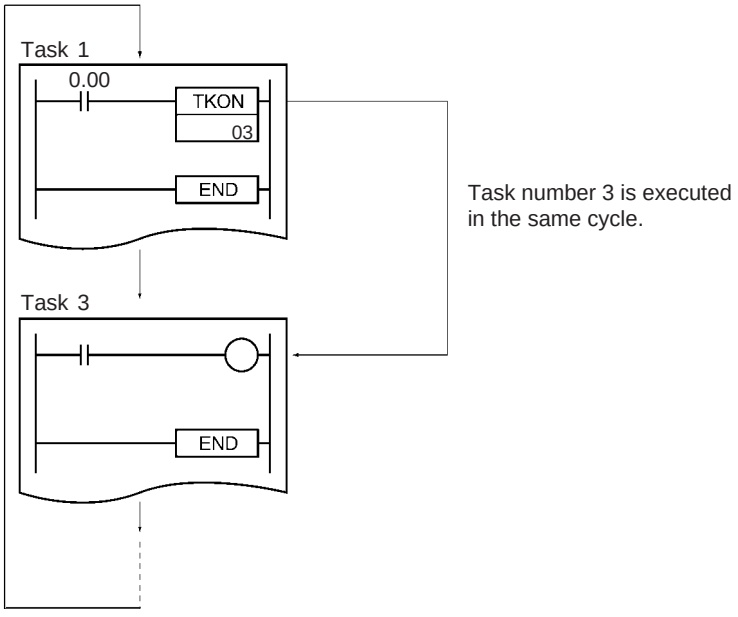
Name	Label	Operation
Error Flag	ER	ON if N is not a constant between 00 and 31 or between 8000 and 8255. ON if the task specified with N does not exist. ON if TKON(820) is executed in an interrupt task. OFF in all other cases.

Name	Addresses	Operation
Task Flags	TK00 to TK31	These flags are turned ON when the corresponding cyclic task is executable and they are OFF when the corresponding cyclic task is not executable or in standby status. TK00 to TK31 correspond to cyclic task numbers 00 to 31.

Examples

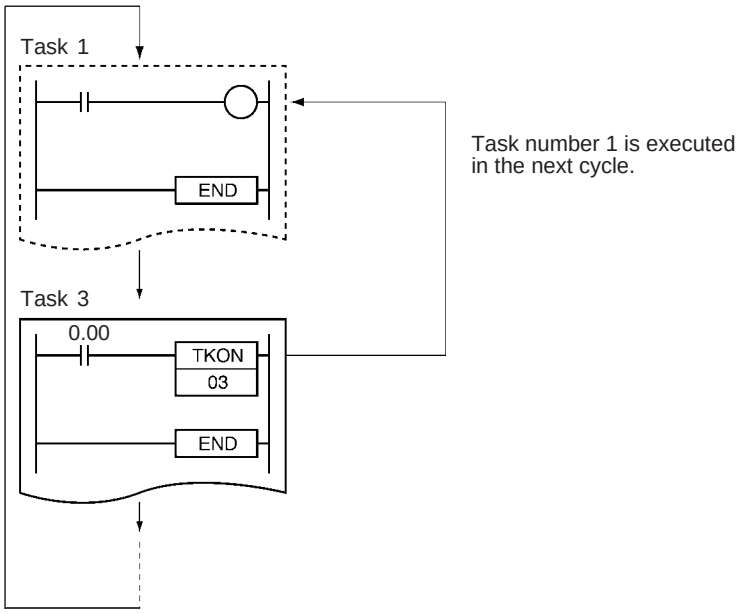
Specifying a Later Task

When CIO 0.00 is ON in the following example, task number 3 is made executable in task number 1. Task number 3 will be executed in the same cycle when program execution reaches task number 3.



Specifying an Earlier Task

When CIO 0.00 is ON in the following example, task number 1 is made executable in task number 3. Task number 1 will be executed in the next cycle when program execution reaches task number 1.

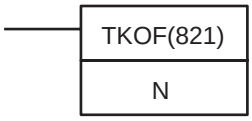


3-32-2 TASK OFF: TKOF(821)

Purpose

Puts the specified cyclic task or extra cyclic task into standby status, i.e., disables execution of the task.

Ladder Symbol



N: Task number

Variations

Variations	**Executed Each Cycle for ON Condition**	TKOF(821)
	Executed Once for Upward Differentiation	@TKOF(821)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Task number

The allowed range for N depends on the kind of task being specified.

- Cyclic tasks:
  N must be a constant between 0 and 31 decimal. (Values 0 to 31 specify cyclic tasks 0 to 31.)
- Extra cyclic tasks:
  N must be a constant between 8000 and 8255 decimal. (Values 8000 to 8255 specify extra cyclic tasks 0 to 255.)

Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	00 to 31 or 8000 to 8255 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

TKOF(821) puts the specified cyclic task or extra cyclic into standby status and turns OFF the corresponding Task Flag (TK00 to TK31).

The task specified in TKOF(821) will be also be in standby status in later cycles as long as it is not put into executable status by TKON(820), the CX-Programmer, or a FINS command.

A task can be put into standby status from any other regular task, although the specified task will not be put into standby status until the next cycle if its task number is lower than the task number of the local task (it would have been executed already). The task will be in standby status in the same cycle if its task number is higher than the local task's task number.

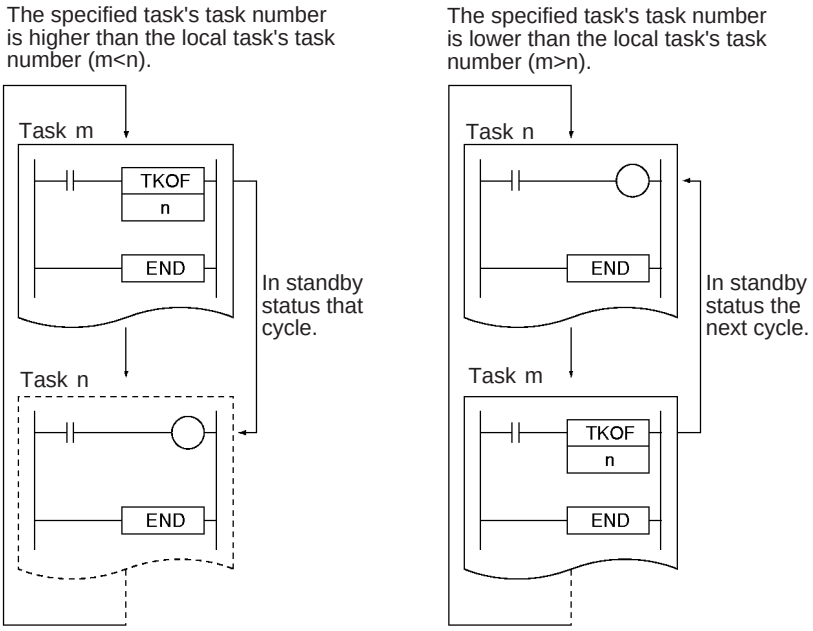
If the local task is specified in TKOF(821), the task will be put into standby status immediately and none of the subsequent instructions in the task will be executed.

Note

(1) The CX-Programmer's *General Properties Tab* for each task has a setting (the *Operation start* box) that specifies whether the cyclic task will be executable at startup. When the *Operation start* box has been checked, the corresponding cyclic task will be put in executable status automatically when the PLC begins operation. All other cyclic tasks will be in non-executable status.

(2) If a task is in non-executable status, TKON(820) can executed to put that task into executable status. Likewise, a cyclic task in executable status can be put into non-executable status with the TKOF(821) instruction.

(3) Cyclic tasks or extra cyclic tasks that are in executable status can be put into standby status by the TKOF(821) instruction.



A regular task that has been set to be executed at startup will be put in executable status automatically when the PLC begins operation. All other regular tasks will be in non-executable status.

A task in executable status can be put in standby status with TKOF(821), the CX-Programmer, or a FINS command.

The terms executable and executing are not interchangeable. Executable tasks are executed in order of their task numbers during cyclic program execution. An executable task will not be executed if it is put in standby status before program execution reaches its task number.

Unlike TKON(820), this instruction can be placed in interrupt tasks as well as in cyclic tasks.

Flags

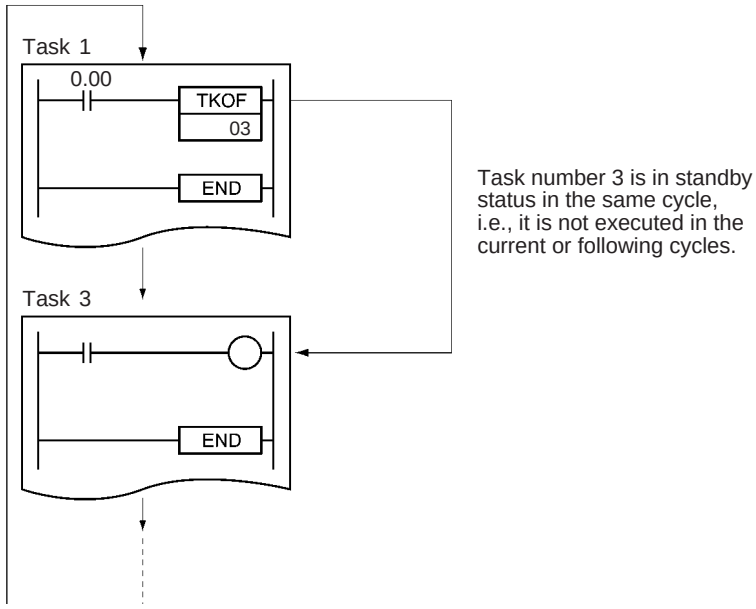
Name	Label	Operation
Error Flag	ER	ON if N is not a constant between 00 and 31 or between 8000 and 8255. ON if the task specified with N does not exist. ON if TKOF(821) is executed in an interrupt task. OFF in all other cases.

Name	Addresses	Operation
Task Flags	TK00 to TK31	These flags are turned ON when the corresponding cyclic task is executable and they are OFF when the corresponding cyclic task is not executable or in standby status. TK00 to TK31 correspond to cyclic task numbers 00 to 31.

Examples

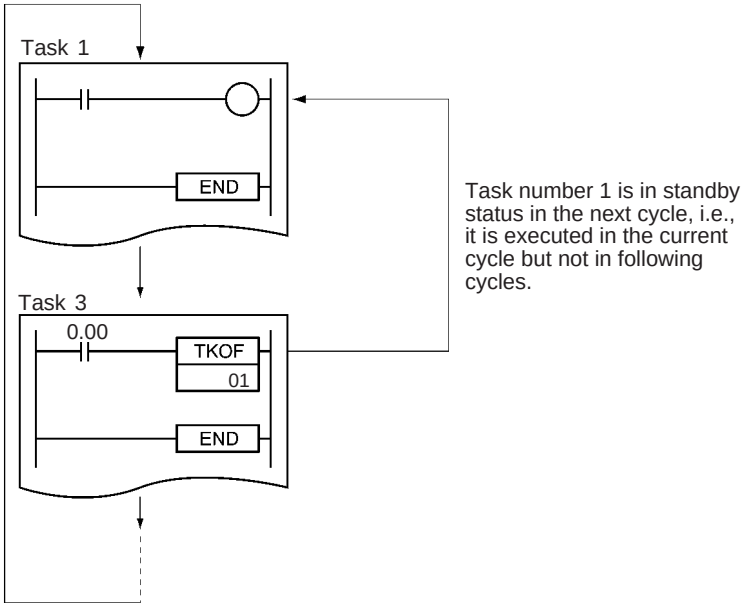
Specifying a Later Task

When CIO 0.00 is ON in the following example, task number 3 is put into standby status in task number 1. Task number 3 will be not be executed in the that cycle when program execution reaches task number 3.



Specifying an Earlier Task

When CIO 0.00 is ON in the following example, task number 1 is put into standby status in task number 3. Task number 1 will be not be executed in the next cycle when program execution reaches task number 1.



3-33 Model Conversion Instructions

This section describes instructions used when changing PLC models.

Instruction	Mnemonic	Function code	Page
BLOCK TRANSFER	XFERC	565	1046
SINGLE WORD DISTRIBUTE	DISTC	566	1048
DATA COLLECT	COLLC	567	1051
MOVE BIT	MOVBC	568	1056
BIT COUNTER	BCNTC	621	1058

The model conversion instructions provide the same functionality as other instructions but use BCD data for the operands, like C-series instructions. (The CP-series use binary data for the operands.) There are five model conversion instructions, as shown in the above table, all of which have a C added to the end of the mnemonic of the equivalent function for binary operand data.

The model conversion instructions enable converting C-series programs to CP-series programs without changing the operand data for these instructions.

When converting C-series programs to CP-series programs on CX-Programmer (see note), these instructions will be automatically used when converting (e.g., XFER will be converted to XFERC), eliminating the need to correct operand data manually.

Note Conversion is achieved by specifying the CP Series as the “device type” in the Change PLC Dialog Box.

Differences from C-series Instructions

“C Series” includes the C200H, C1000H, C2000H, C200HS, C2000HX/HG/HE(-Z), CQM1, CQM1H, CPM1/CPM1A, CPM2C, and SRM1.

Name	Model conversion instruction	Corresponding C-series instruction	Differences from C-series instructions		When converting device type to CP with CX-Programmer
	Mnemonic (function code)	Mnemonic (function code)	C200H, C1000H, or C2000H	C200HS, C2000HX/HG/HE(-Z), CQM1, CQM1H, CPM1/CPM1A, CPM2C, or SRM1	
BLOCK TRANSFER	XFERC(565)	XFER(70)	Same	Same	XFER is converted to XFERC. Operands do not require correction.
SINGLE WORD DISTRIBUTE	DISTC(566)	DIST(80)	Along with data distribution operation, provides stack push operation not previously supported.	Same (distribution operation and stack push operation)	DIST is converted to DICTC. Operands do not require correction.
DATA COLLECT	COLLC(567)	COLL(81)	Along with data collection operation, provides stack read operation not previously supported.	Same (data collection operation and stack read operation)	COLL is converted to COLLC. Operands do not require correction.
MOVE BIT	MOVBC(568)	MOVB(82)	Same	Same	MOVB is converted to MOVBC. Operands do not require correction.
BIT COUNTER	BCNTC(621)	BCNT(67)	Same	Same	BCNT is converted to BCNTC. Operands do not require correction.

Note The operation of the Conditions Flags differs in the following ways. Refer to the description of the Conditions Flags for each instruction for details.

- The operation of the Conditions Flags differs for all instructions when the contents of a DM Area words used for indirect addressing is not BCD (*BCD) or the DM Area addressing range is exceeded.
- For DISTC(566), the operation of the Conditions Flags differs in comparison with that for the C200H, C1000H, and C2000H for the stack push operation.
- For COLLC(567), the operation of the Conditions Flags differs in comparison with that for the C200H, C1000H, and C2000H for the stack read operation.

Differences from Standard CP-series Instructions

Name	Model conversion instruction	Corresponding standard CP-series instruction	Differences from standard CP-series instructions (See note.)
	Mnemonic (function code)	Mnemonic (function code)	
BLOCK TRANSFER	XFERC(565)	XFER(70)	The data type for the first operand (number of words to transfer) is BCD (0000 to 9999) instead of binary (0000 to FFFF hex).
SINGLE WORD DISTRIBUTE	DISTC(566)	DIST(80)	A stack push operation is supported in addition to the data distribution operation. The data type for the third operand (offset data) is BCD (data distribution: 0000 to 7999, stack push: 0000 to 9999) instead of binary (0000 to FFFF hex).
DATA COLLECT	COLLC(567)	COLL(81)	A stack read operation is supported in addition to the data distribution operation. The data type for the second operand (offset data) is BCD (data distribution: 0000 to 7999, stack read for FIFO: 9000 to 9999, stack read for LIFO: 8000 to 8999) instead of binary (0000 to FFFF hex).
MOVE BIT	MOVBC(568)	MOVB(82)	The data type for the source and destination bit specifications in the second operand (control data) is BCD (00 to 15) instead of binary (00 to 0F hex).
BIT COUNTER	BCNTC(621)	BCNT(67)	The data type for the first operand (number of words to count) is BCD (0000 to 9999) instead of binary (0000 to FFFF hex). The data type stored for the third operand (count results) is BCD (0000 to 9999) instead of binary (0000 to FFFF hex).

Note The operation of the Conditions Flags differs in the following ways. Refer to the description of the Conditions Flags for each instruction for details.

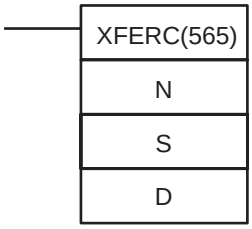
- The Error Flag will turn ON if the data for the above operands is not BCD.
- For DISTC(566), the operation of the Conditions Flags was added for the stack push operation.
- For COLLC(567), the operation of the Conditions Flags was added for the stack read operation.

3-33-1 BLOCK TRANSFER: XFERC(565)

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	XFERC(565)
	Executed Once for Upward Differentiation	@XFERC(565)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

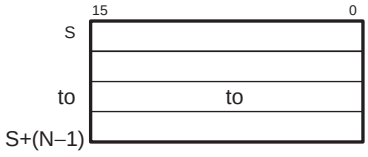
Operands

N: Number of Words

Specifies the number of words to be transferred. The possible range for N is 0000 to 9999 BCD.

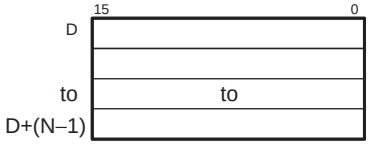
S: First Source Word

Specifies the first source word.



D: First Destination Word

Specifies the first destination word.



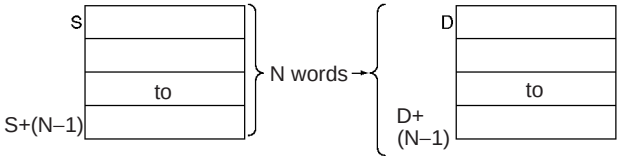
Operand Specifications

Area	N	S	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		

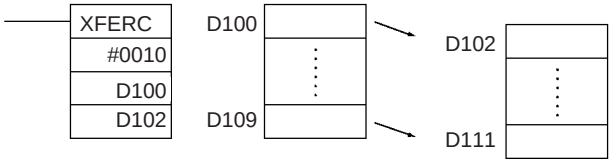
Area	N	S	D
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #9999 (BCD)	---	---
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

XFERC(565) 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 XFERC(565) can perform word-shift operations.



Flags

Name	Label	Operation
Error Flag	ER	ON if the data in N (the number of words) is not BCD.

Note In C-series PLCs, the BLOCK TRANSFER (XFER) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. XFERC(565) will not cause the Error Flag to go ON in these cases.

Precautions

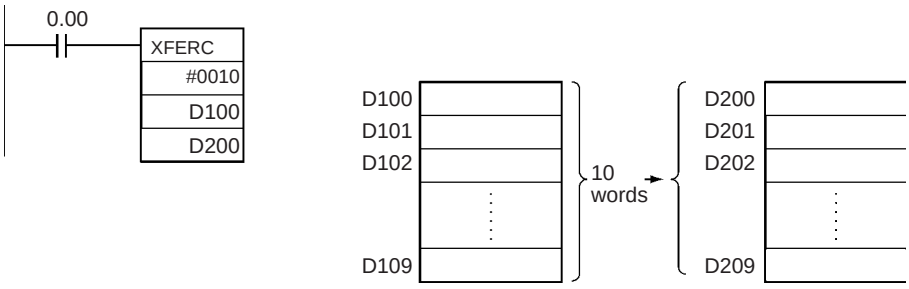
Be sure that the source words (S to S+N–1) and destination words (D to D+N–1) do not exceed the end of the data area.

Some time will be required to complete XFERC(565) when a large number of words is being transferred. In this case, the XFERC(565) transfer might not be completed if a power interruption occurs during execution of the instruction.

The content of N must be BCD. If N is not BCD, an error will occur and the Error Flag will be turned ON.

Example

When CIO 0.00 is ON in the following example, the 10 words D100 through D109 are copied to D200 through D209.

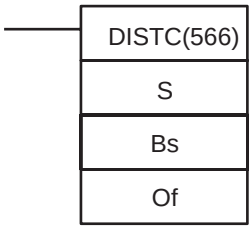


3-33-2 SINGLE WORD DISTRIBUTE: DISTC(566)

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	DISTC(566)
	Executed Once for Upward Differentiation	@DISTC(566)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

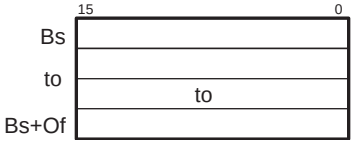
Bs: Destination Base Address

Specifies the destination base address. The offset is added to this address to calculate the destination word.

Of: Offset

- Data Distribution Operation (0000 to 7999 BCD)

  This value is added to the base address to calculate the destination word. The offset can be any value from 0000 to 7999 in BCD, but Bs and Bs+Of must be in the same data area.



- Stack Push Operation (9000 to 9999 BCD)

  When the leftmost digit of Of is 9, the rightmost 3 digits of Of specify the number of words in the stack. The offset can be any value from 9000 to 9999 BCD.

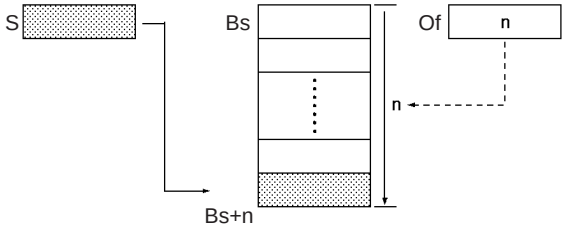
Operand Specifications

Area	S	Bs	Of
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959	A448 to A959	A0 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	---	#0000 to #7999 for distribution #9000 to #9999 for stack operation
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

Data Distribution Operation

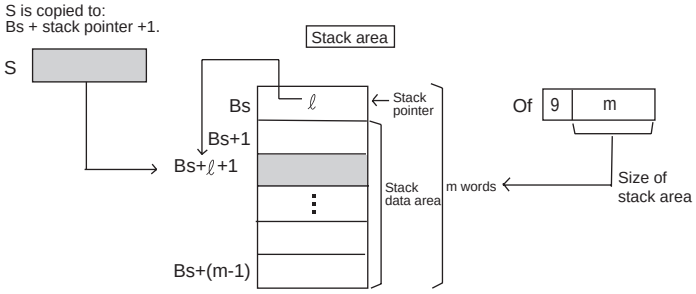
DISTC(566) copies S to the destination word calculated by adding Of to Bs. The same DISTC(566) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.



Stack Push Operation

When the leftmost digit (bits 12 to 15) of Of is 9 BCD, DISTC(566) operates a stack from Bs to Bs+Of-9000. The destination base address (Bs) contains the stack pointer and the rest of the words in the stack contain the stack data.

DISTC(566) copies S to the destination word calculated by adding the stack pointer (content of Bs) + 1 to address Bs. The same DISTC(566) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.



Each time that the content of S is copied to a word in the stack data area, the stack pointer in Bs is automatically incremented by +1.

Note Use COLLC(567) to read stack data from the stack area.

Flags

Name	Label	Operation
Error Flag	ER	ON if Stack Push Operation is specified, but the stack pointer data in Bs is not BCD. ON if Stack Push Operation is specified and the stack pointer indicates a word that exceeds the stack data area.
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.

Note In C-series PLCs, the SINGLE WORD DISTRIBUTE (DIST) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. DISTC(566) will not cause the Error Flag to go ON in these cases.

Precautions

Once DISTC(566) has been executed with Stack Push Operation to allocate a stack area, always specify the same length stack area in subsequent DISTC(566) instructions. Operation will be unreliable if a different stack area size is specified in later DISTC(566) instructions.

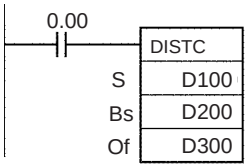
Be sure that the offset or stack size specified by Of does not exceed the end of the data area when added to Bs.

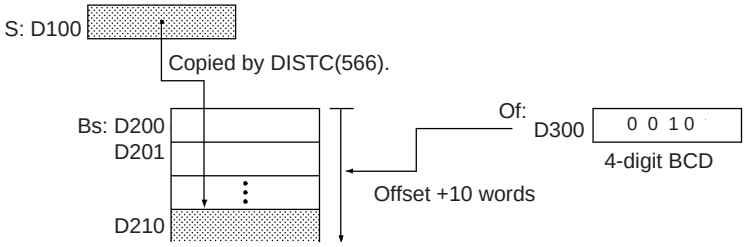
Examples

Data Distribution Operation

The leftmost byte of D300 is 0, so DISTC(566) performs the Data Distribution Operation.

When CIO 0.00 is ON in the following example, the contents of D100 will be copied to D210 (D200 + 10) if the content of D300 is 0010 BCD. The content of D100 can be copied to other words by changing the offset in D300.

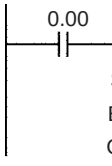


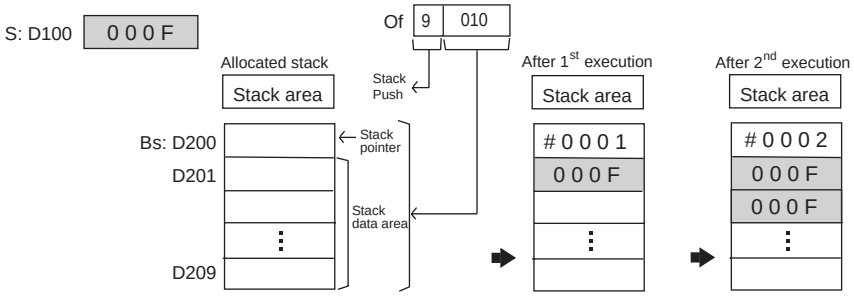


Stack Push Operation

The leftmost byte of Of is 9, so DISTC(566) performs the Stack Push Operation.

When CIO 0.00 is ON in the following example, DISTC(566) allocates a 10 word stack area (since the rightmost 3 digits of Of are #010) between D200 and D209. At the same time, the contents of D100 will be copied to the word calculated by adding D200 + stack pointer +1. Finally, the stack pointer is incremented by +1.



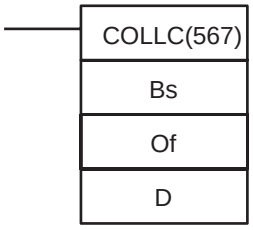


3-33-3 DATA COLLECT: COLLC(567)

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	COLLC(567)
	Executed Once for Upward Differentiation	@COLLC(567)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

Bs: Source Base Address

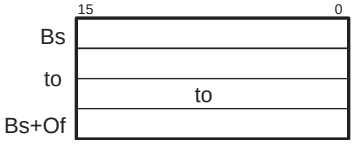
Specifies the source base address. The offset is added to this address to calculate the source word.

Of: Offset

The value of Of determines the operation of COLLC(567).

- Data Collect Operation (Of = 0000 to 7999 BCD)

  The Of value is added to the base address to calculate the source word. The offset can be any value from 0000 to 7999 BCD, but Bs and Bs+Of must be in the same data area.



- LIFO Stack Read Operation (Of = 8000 to 8999 BCD)
  If the leftmost digit of Of is 8, COLLC(567) will operate as a LIFO stack instruction. The stack begins at Bs with a length specified in the rightmost 3 digits of Of.
- FIFO Stack Read Operation (Of = 9000 to 9999 BCD)
  If the leftmost digit of Of is 9, COLLC(567) will operate as a FIFO stack instruction. The stack begins at Bs with a length specified in the rightmost 3 digits of Of.

Operand Specifications

Area	Bs	Of	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	---	#0000 to #7999 for Data Collection #8000 to #8999 for LIFO Stack Read #9000 to #9999 for FIFO Stack Read	---
Data Registers	---	DR0 to DR15	

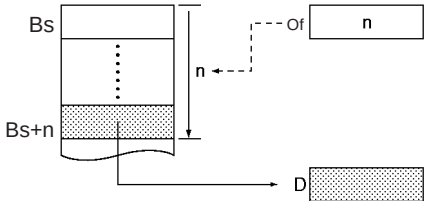
Area	Bs	Of	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

Depending on the value of Of, COLLC(567) will operate as a data collection instruction, FIFO stack instruction, or LIFO stack instruction.

Data Collection Operation (Of = 0000 to 7999 BCD)

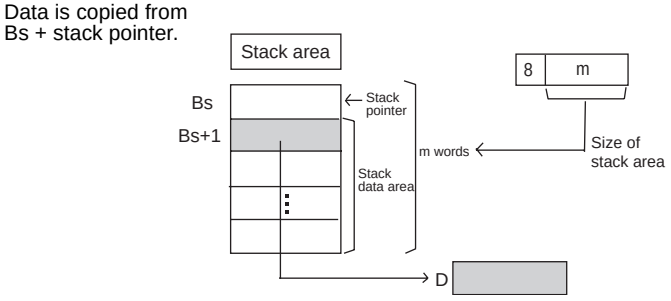
COLLC(567) copies the source word (calculated by adding Of to Bs) to the destination word. The same COLLC(567) instruction can be used to collect data from various source words in the data area by changing the value of Of.



LIFO Stack Read Operation (Of = 8000 to 8999 BCD)

If the leftmost digit of Of is 8, COLLC(567) will operate as a LIFO stack instruction (LIFO stands for Last-In-First-Out). In this case, the rightmost 3 digits of Of specify the size of the stack.

COLLC(567) copies the data most recently recorded in the stack to D. The source word is Bs + the stack pointer (content of Bs). After the data is copied, the stack pointer is decremented by 1.

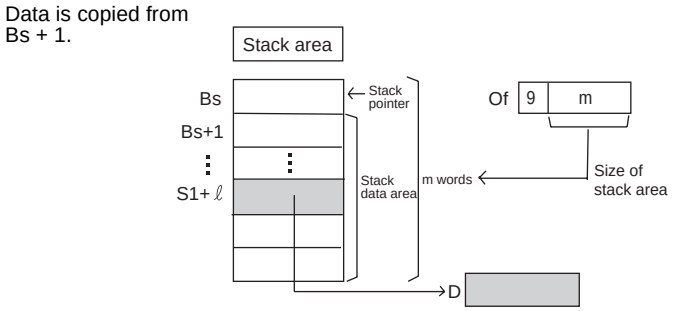


Note Use DISTC(566) to write stack data to the stack area.

FIFO Stack Read Operation (Of = 9000 to 9999 BCD)

If the leftmost digit of Of is 9, COLLC(567) will operate as a FIFO stack instruction (FIFO stands for First-In-First-Out). In this case, the rightmost 3 digits of Of specify the size of the stack.

COLLC(567) copies the data from the oldest word recorded in the stack to D. The source word is Bs + 1. After the data is copied, the stack pointer is decremented by 1.



Note Use DISTC(566) to write stack data to the stack area.

Flags

Name	Label	Operation
Error Flag	ER	ON if the offset data in Of is not BCD. ON if LIFO or FIFO Stack Operation is specified, but the stack pointer data in Bs is not BCD. ON if LIFO or FIFO Stack Operation is specified and the stack pointer indicates a word that exceeds the stack data area. OFF in all other cases.
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.

Note In C-series PLCs, the DATA COLLECT (COLL) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. COLLC(567) will not cause the Error Flag to go ON in these cases.

Precautions

Once DISTC(566) has been executed with Stack Push Operation to allocate a stack area, always specify that same length stack area in the COLLC(567) instructions. Operation will be unreliable if a different stack area size is specified in the COLLC(567) instructions.

Be sure that the offset or stack size specified by Of does not exceed the end of the data area when added to Bs.

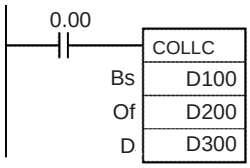
The offset data in Of must be BCD.

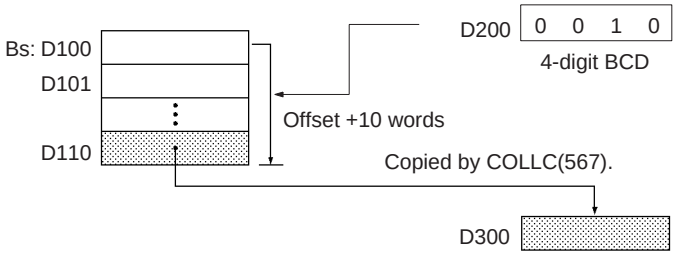
Examples

Data Collection Operation

The leftmost byte of D200 is 0, so COLLC(567) performs the Data Collection Operation.

When CIO 0.00 is ON in the following example, the contents of D110 (D100 + 10) will be copied to D300 if the content of D200 is 10 (0010 BCD). The contents of other words can be copied to D300 by changing the offset in D200.

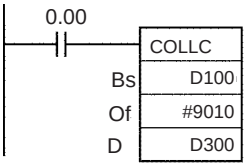


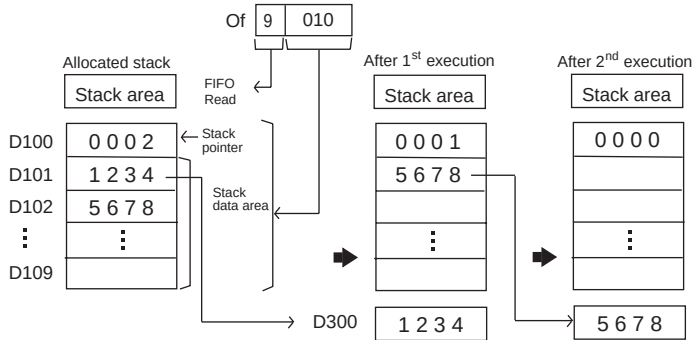


FIFO Stack Operation

The leftmost byte of Of is 9, so COLLC(567) performs the FIFO Stack Operation.

When CIO 0.0 is ON in the following example, COLLC(567) allocates a 10 word stack area (since the rightmost 3 digits of Of are #010) between D100 and D109. At the same time, the contents of D101 (Bs +1) are copied to D300. Finally, the stack pointer is decremented by 1.

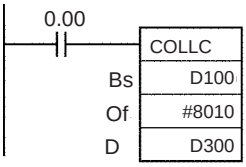


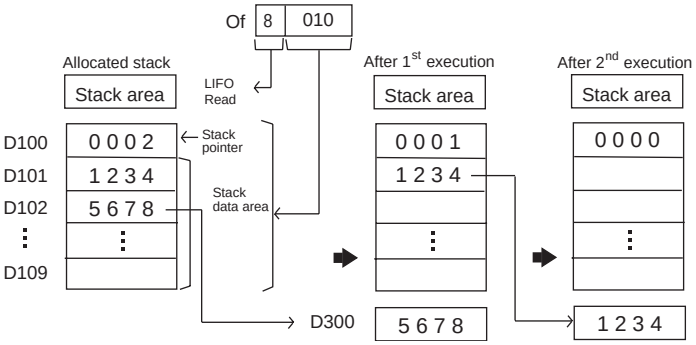


LIFO Stack Operation

The leftmost byte of Of is 8, so COLLC(567) performs the LIFO Stack Operation.

When CIO 0.00 is ON in the following example, COLLC(567) allocates a 10 word stack area (since the rightmost 3 digits of Of are #010) between D100 and D109. At the same time, the contents of the source word (D100 + stack pointer) are copied to D300. Finally, the stack pointer is decremented by 1.



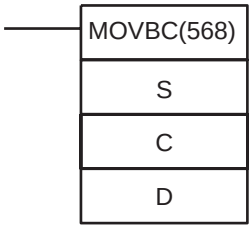


3-33-4 MOVE BIT: MOVBC(568)

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	MOVBC(568)
	Executed Once for Upward Differentiation	@MOVBC(568)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

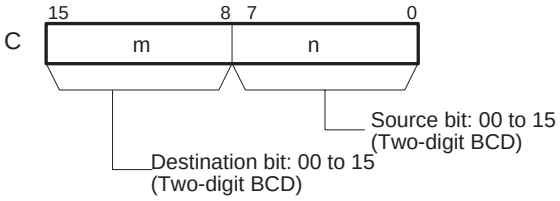
Applicable Program Areas

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.



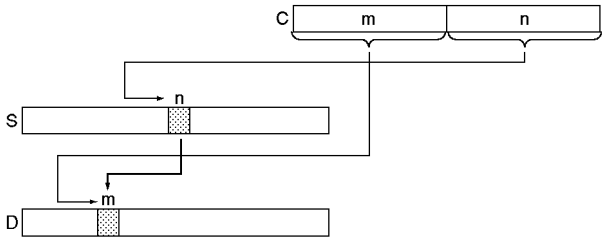
Operand Specifications

Area	S	C	D
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Registers	DR0 to DR15		

Area	S	C	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

MOVBC(568) 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.

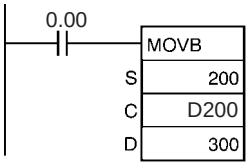
Flags

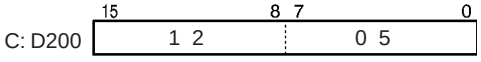
Name	Label	Operation
Error Flag	ER	ON if the rightmost and leftmost two digits of C are not BCD or outside of the specified range of 00 to 15. OFF in all other cases.

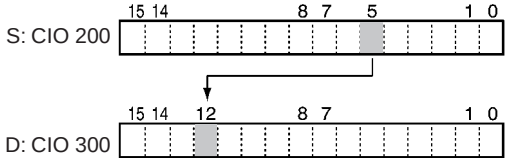
Note In C-series PLCs, the MOVE BIT (MOVB) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. MOVBC(568) will not cause the Error Flag to go ON in these cases.

Examples

When CIO 0.00 is ON in the following example, the 5th bit of the source word (CIO 200) is copied to the 12th bit of the destination word (CIO 300) in accordance with the control word's value of 1205.





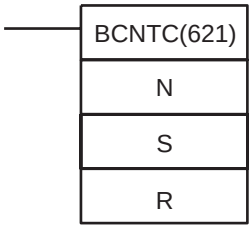


3-33-5 BIT COUNTER: BCNTC(621)

Purpose

Counts the total number of ON bits in the specified word(s).

Ladder Symbol



N: Number of words

S: First source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCNTC(621)
	Executed Once for Upward Differentiation	@BCNTC(621)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		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 9999 (BCD).

S: First source word

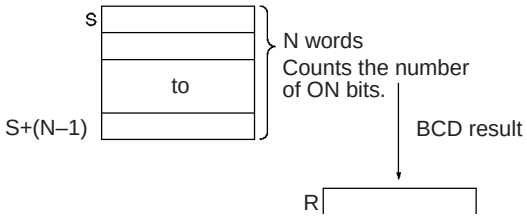
S and S+(N–1) must be in the same data area.

Operand Specifications

Area	N	S	R
CIO Area	CIO 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 to *D32767		
Constants	#0001 to #9999 (BCD)	---	
Data Registers	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 IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BCNTC(621) counts the total number of bits that are ON in all words between S and S+(N–1) and places the BCD result in R.



Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the range 0001 to 9999 BCD. ON if result exceeds 9999 BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

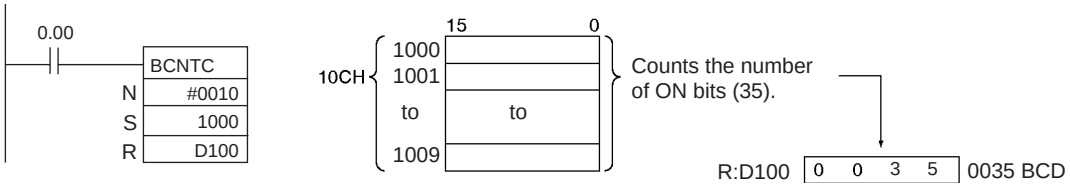
Note In C-series PLCs, the BIT COUNTER (BITC) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. BCNTC(621) will not cause the Error Flag to go ON in these cases.

Precautions

An error will occur if N is not BCD between 0001 and 9999, or the result exceeds 9,999.

Example

When CIO 0.00 is ON in the following example, BCNTC(621) counts the total number of ON bits in the 10 words from CIO 1000 through CIO 1009 and writes the result to D100.

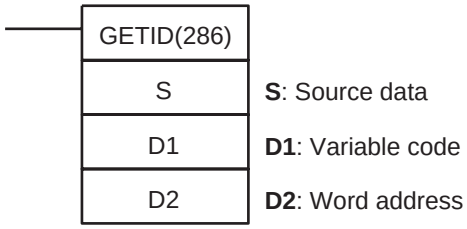


3-33-6 GET VARIABLE ID: GETID(286)

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.
Immediate Refreshing Specification		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 0 to CIO 6143		
Work Area	W0 to W511		
Holding Bit Area	H0 to H511		
Auxiliary Bit Area	A0 to A959		
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0 to D32767		
Indirect DM addresses in binary	@ D0 to @ D32767		
Indirect DM addresses in BCD	*D0 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 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 6143)
Work Area	W		00B1 hex	0000 to 01FF hex (000 to 511)
Holding Bit Area	H		00B2 hex	0000 to 01FF hex (000 to 511)
Auxiliary Bit Area			00B3 hex	0000 to 03BF hex (000 to 959)
DM Area			0082 hex	0000 to 7FFF hex (00000 to 32767)

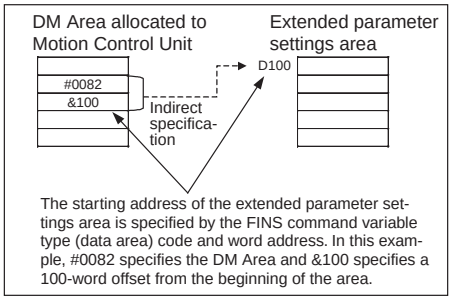
Variables in function blocks are automatically allocated addresses by CX-Programmer Ver. 5.0 and later systems, unless the AT specification is used. For example, if it is necessary to indirectly specify the extended parameter settings of a Special Unit such as a Motion Control Unit and a variable is 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

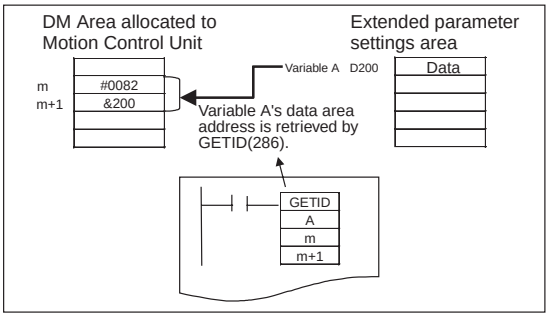
Name	Label	Operation
Error Flag	ER	ON if S is not within the allowed range.

Example

Normal Operation



Using Function Blocks



SECTION 4
Instruction Execution Times and Number of Steps

This section lists the execution times and number of steps for all instructions supported by the CP1H PLCs, and describes the execution times for function block instances.

4-1 Instruction Execution Times and Number of Steps

The following table lists the execution times for all instructions that are available for CP PLCs.

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 depend the conditions when the instruction is executed. The execution time is also required 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 CP-series 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 is measured in steps. Basically speaking, 1 step is equivalent to 1 word. (Program capacity for previous OMRON PLCs was measured in words.)

(2) Most instructions are supported in differentiated form (indicated with ↑, ↓, @, and %). Specifying differentiation will increase the execution times by the following amounts.

Symbol	CP1H CPU Unit
↑ or ↓	+0.24 μs
@ or %	+0.24 μs

(3) Use the following times as guidelines when instructions are not executed.
CP1H CPU Unit: Approx. 0.1 μs

(4) When converting programs from previous models of PLC (C-series or CV/CVM1-series) to a CP-series PLC, the following guidelines can be used to convert the program size from words to steps. Add the values (n) given in the following table for each instruction to calculate the CP-series program size in steps. Conversion is not necessary when converting programs from CS/CJ-series PLCs (i.e., the sizes will not change).

Previous size in “a” words - CP1H size in “a+n” steps			
Instruction type	**Instruction options**	**Value of “n” when converting C-series PLC to CP-series PLC**	**Value of “n” when converting CV/CVM1-series PLC to CP-series PLC**
Basic instructions	None	–1 for OUT, SET, RESET, and KEEP 0 for all other instructions	0
	Upward differentiation	0	+1
	Immediate refreshing	---	0
	Upward differentiation and immediate refreshing	---	+2

Previous size in “a” words - CP1H size in “a+n” steps			
Instruction type	**Instruction options**	**Value of “n” when converting C-series PLC to CP-series PLC**	**Value of “n” when converting CV/CVM1-series PLC to CP-series PLC**
Special instructions	None	0	–1
	Upward differentiation	+1	0
	Immediate refreshing	---	+3
	Upward differentiation and immediate refreshing	---	+4

Examples:
For a C-series PLC, the OUTPUT instruction requires 2 words per instruction, so 1 step (2 – 1) would be required for a CP-series PLC.
For a CV/CVM1-series PLC, the immediate refresh variation of the MOVE instruction (!MOV) requires 4 words per instruction, so 7 steps (4 + 3) would be required for a CP-series PLC.

Sequence Input Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
LOAD	LD	---	1	0.10	---
	!LD	---	2	+24.10	Increase for immediate refresh
LOAD NOT	LD NOT	---	1	0.10	---
	!LD NOT	---	2	+24.10	Increase for immediate refresh
AND	AND	---	1	0.10	---
	!AND	---	2	+24.10	Increase for immediate refresh
AND NOT	AND NOT	---	1	0.10	---
	!AND NOT	---	2	+24.10	Increase for immediate refresh
OR	OR	---	1	0.10	---
	!OR	---	2	+24.10	Increase for immediate refresh
OR NOT	OR NOT	---	1	0.10	---
	!OR NOT	---	2	+24.10	Increase for immediate refresh
AND LOAD	AND LD	---	1	0.05	---
OR LOAD	OR LD	---	1	0.05	---
NOT	NOT	520	1	0.05	---
CONDITION ON	UP	521	3	0.50	---
CONDITION OFF	DOWN	522	4	0.50	---
LOAD BIT TEST	LD TST	350	4	0.35	---
LOAD BIT TEST NOT	LD TSTN	351	4	0.35	---
AND BIT TEST NOT	AND TSTN	351	4	0.35	---
OR BIT TEST	OR TST	350	4	0.35	---
OR BIT TEST NOT	OR TSTN	351	4	0.35	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table

Sequence Output Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
OUTPUT	OUT	---	1	0.35	---
	!OUT	---	2	+23.07	Increase for immediate refresh
OUTPUT NOT	OUT NOT	---	1	0.35	---
	!OUT NOT	---	2	+23.07	Increase for immediate refresh
KEEP	KEEP	11	1	0.40	---
DIFFERENTIATE UP	DIFU	13	2	0.50	---
DIFFERENTIATE DOWN	DIFD	14	2	0.50	---
SET	SET	---	1	0.30	---
	!SET	---	2	+23.17	Increase for immediate refresh
RESET	RSET	---	1	0.30	Word specified
	!RSET	---	2	+23.17	Increase for immediate refresh
MULTIPLE BIT SET	SETA	530	4	11.77	With 1-bit set
				67.03	With 1,000-bit set
MULTIPLE BIT RESET	RSTA	531	4	11.8	With 1-bit reset
				69.63	With 1,000-bit reset
SINGLE BIT SET	SETB	532	2	0.5	---
	!SETB		3	+23.31	---
SINGLE BIT RESET	RSTB	533	2	0.5	---
	!RSTB		3	+23.31	---
SINGLE BIT OUTPUT	OUTB	534	2	0.45	---
	!OUTB		3	+23.22	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Sequence Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
END	END	1	1	9.18	---
NO OPERATION	NOP	0	1	0.05	---
INTERLOCK	IL	2	1	0.15	---
INTERLOCK CLEAR	ILC	3	1	0.15	---
MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	517	3	10.3	During interlock
				13.3	Not during interlock and interlock not set
				16.6	Not during interlock and interlock set
MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	518	3	10.3	During interlock
				13.3	Not during interlock and interlock not set
				16.6	Not during interlock and interlock set
MULTI-INTERLOCK CLEAR	MILC	519	2	8.3	Interlock not cleared
				9.6	Interlock cleared
JUMP	JMP	4	2	0.95	---
JUMP END	JME	5	2	---	---
CONDITIONAL JUMP	CJP	510	2	0.95	When JMP condition is satisfied

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
CONDITIONAL JUMP NOT	CJPN	511	2	0.95	When JMP condition is satisfied
MULTIPLE JUMP	JMP0	515	1	0.15	---
MULTIPLE JUMP END	JME0	516	1	0.15	---
FOR LOOP	FOR	512	2	1.00	Designating a constant
BREAK LOOP	BREAK	514	1	0.15	---
NEXT LOOP	NEXT	513	1	0.45	When loop is continued
				0.55	When loop is ended

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Timer and Counter Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
TIMER	TIM	---	3	1.30	---
	TIMX	550			
COUNTER	CNT	---	3	1.30	---
	CNTX	546			
HIGH-SPEED TIMER	TIMH	15	3	1.80	---
	TIMHX	551			
ONE-MS TIMER	TMHH	540	3	1.75	---
	TMHHX	552			
ACCUMULATIVE TIMER	TTIM	87	3	24.81	---
				17.79	When resetting
				13.97	When interlocking
	TTIMX	555		23.78	---
				17.76	When resetting
				14.11	When interlocking
LONG TIMER	TIML	542	4	15.69	---
				13.61	When interlocking
	TIMLX	553		17.51	---
				13.11	When interlocking
MULTI-OUTPUT TIMER	MTIM	543	4	35.36	---
				12.81	When resetting
	MTIMX	554		41.95	---
				17.42	When resetting
REVERSIBLE COUNTER	CNTR	12	3	29.03	---
	CNTRX	548		22.44	
RESET TIMER/ COUNTER	CNR	545	3	15.27	When resetting 1 word
				5.95 ms	When resetting 1,000 words
	CNRX	547		14.44	When resetting 1 word
				5.95 ms	When resetting 1,000 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Comparison Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
Input Comparison Instructions (unsigned)	LD, AND, OR +=	300	4	0.35	---
	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	---
	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			---
Input Comparison Instructions (signed)	LD, AND, OR +=+S	302	4	0.35	---
	LD, AND, OR +<>+S	307			
	LD, AND, OR +<+S	312			
	LD, AND, OR +<=	317			
	LD, AND, OR +>+S	322			
	LD, AND, OR +>=+S	327			
Input Comparison Instructions (double, signed)	LD, AND, OR +=+SL	303	4	0.35	---
	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			
Time Comparison Instructions	LD, AND, OR +DT	341	4	18.8	---
	LD, AND, OR +<>DT	342	4	45.6	---
	LD, AND, OR +<DT	343	4	45.6	---
	LD, AND, OR +<=DT	344	4	18.8	---
	LD, AND, OR +>DT	345	4	45.6	---
	LD, AND, OR +>=DT	346	4	18.8	---
COMPARE	CMP	20	3	0.10	---
	!CMP	20	7	+45.2	Increase for immediate refresh
DOUBLE COMPARE	CMPL	60	3	0.50	---
SIGNED BINARY COMPARE	CPS	114	3	0.30	---
	!CPS	114	7	+45.2	Increase for immediate refresh
DOUBLE SIGNED BINARY COMPARE	CPSL	115	3	0.50	---
TABLE COMPARE	TCMP	85	4	27.66	---
MULTIPLE COMPARE	MCMP	19	4	42.33	---
UNSIGNED BLOCK COMPARE	BCMP	68	4	47.21	---
EXPANDED BLOCK COMPARE	BCMP2	502	4	13.20	Number of data words: 1
				650.0	Number of data words: 255

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
AREA RANGE COMPARE	ZCP	88	3	11.53	---
DOUBLE AREA RANGE COMPARE	ZCPL	116	3	11.28	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Data Movement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
MOVE	MOV	21	3	0.30	---
	!MOV	21	7	+35.1	Increase for immediate refresh
DOUBLE MOVE	MOVL	498	3	0.60	---
MOVE NOT	MVN	22	3	0.35	---
DOUBLE MOVE NOT	MVNL	499	3	0.60	---
MOVE BIT	MOVB	82	4	0.50	---
MOVE DIGIT	MOVD	83	4	0.50	---
MULTIPLE BIT TRANSFER	XFRB	62	4	20.1	Transferring 1 bit
				266.30	Transferring 255 bits
BLOCK TRANSFER	XFER	70	4	8.80	Transferring 1 word
				1.18 ms	Transferring 1,000 words
BLOCK SET	BSET	71	4	14.63	Setting 1 word
				570.17	Setting 1,000 words
DATA EXCHANGE	XCHG	73	3	0.80	---
DOUBLE DATA EXCHANGE	XCGL	562	3	1.5	---
SINGLE WORD DISTRIBUTE	DIST	80	4	12.77	---
DATA COLLECT	COLL	81	4	12.85	---
MOVE TO REGISTER	MOVR	560	3	0.60	---
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 following table.

Data Shift Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SHIFT REGISTER	SFT	10	3	12.68	Shifting 1 word
				1.49 ms	Shifting 1,000 words
REVERSIBLE SHIFT REGISTER	SFTR	84	4	13.76	Shifting 1 word
				1.54 ms	Shifting 1,000 words
ASYNCHRONOUS SHIFT REGISTER	ASFT	17	4	14.21	Shifting 1 word
				2.94 ms	Shifting 1,000 words
WORD SHIFT	WSFT	16	4	11.20	Shifting 1 word
				1.47 ms	Shifting 1,000 words

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
ARITHMETIC SHIFT LEFT	ASL	25	2	0.45	---
DOUBLE SHIFT LEFT	ASLL	570	2	0.80	---
ARITHMETIC SHIFT RIGHT	ASR	26	2	0.45	---
DOUBLE SHIFT RIGHT	ASRL	571	2	0.80	---
ROTATE LEFT	ROL	27	2	0.45	---
DOUBLE ROTATE LEFT	ROLL	572	2	0.80	---
ROTATE LEFT WITHOUT CARRY	RLNC	574	2	0.45	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	2	0.80	---
ROTATE RIGHT	ROR	28	2	0.45	---
DOUBLE ROTATE RIGHT	RORL	573	2	0.80	---
ROTATE RIGHT WITHOUT CARRY	RRNC	575	2	0.45	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	2	0.80	---
ONE DIGIT SHIFT LEFT	SLD	74	3	11.86	Shifting 1 word
				1.24 ms	Shifting 1,000 words
ONE DIGIT SHIFT RIGHT	SRD	75	3	13.95	Shifting 1 word
				1.85 ms	Shifting 1,000 words
SHIFT N-BIT DATA LEFT	NSFL	578	4	14.39	Shifting 1 bit
				90.10	Shifting 1,000 bits
SHIFT N-BIT DATA RIGHT	NSFR	579	4	14.43	Shifting 1 bit
				130.27	Shifting 1,000 bits
SHIFT N-BITS LEFT	NASL	580	3	0.45	---
DOUBLE SHIFT N-BITS LEFT	NSLL	582	3	0.80	---
SHIFT N-BITS RIGHT	NASR	581	3	0.45	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	3	0.80	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Increment/Decrement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
INCREMENT BINARY	++	590	2	0.45	---
DOUBLE INCREMENT BINARY	++L	591	2	0.80	---
DECREMENT BINARY	– –	592	2	0.45	---
DOUBLE DECREMENT BINARY	– –L	593	2	0.80	---
INCREMENT BCD	++B	594	2	12.09	---
DOUBLE INCREMENT BCD	++BL	595	2	10.59	---
DECREMENT BCD	– –B	596	2	11.63	---
DOUBLE DECREMENT BCD	– –BL	597	2	9.59	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Symbol Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SIGNED BINARY ADD WITHOUT CARRY	+	400	4	0.30	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	4	0.60	---
SIGNED BINARY ADD WITH CARRY	+C	402	4	0.40	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	4	0.60	---
BCD ADD WITHOUT CARRY	+B	404	4	18.14	---
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	4	22.87	---
BCD ADD WITH CARRY	+BC	406	4	19.7	---
DOUBLE BCD ADD WITH CARRY	+BCL	407	4	23.63	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	4	0.3	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	4	0.60	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	4	0.40	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	4	0.60	---
BCD SUBTRACT WITHOUT CARRY	–B	414	4	17.57	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	4	22.09	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
BCD SUBTRACT WITH CARRY	–BC	416	4	18.37	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	4	22.91	---
SIGNED BINARY MULTIPLY	*	420	4	0.65	---
DOUBLE SIGNED BINARY MULTIPLY	*L	421	4	13.02	---
UNSIGNED BINARY MULTIPLY	*U	422	4	0.75	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	4	13.23	---
BCD MULTIPLY	*B	424	4	16.83	---
DOUBLE BCD MULTIPLY	*BL	425	4	33.33	---
SIGNED BINARY DIVIDE	/	430	4	0.70	---
DOUBLE SIGNED BINARY DIVIDE	/L	431	4	13.35	---
UNSIGNED BINARY DIVIDE	/U	432	4	0.8	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	4	12.91	---
BCD DIVIDE	/B	434	4	18.03	---
DOUBLE BCD DIVIDE	/BL	435	4	27.77	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Conversion Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
BCD-TO-BINARY	BIN	023	3	0.40	---
DOUBLE BCD-TO-DOUBLE BINARY	BINL	058	3	10.41	---
BINARY-TO-BCD	BCD	024	3	10.22	---
DOUBLE BINARY-TO-DOUBLE BCD	BCDL	059	3	10.18	---
2'S COMPLEMENT	NEG	160	3	0.35	---
DOUBLE 2'S COMPLEMENT	NEGL	161	3	0.60	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN	600	3	0.60	---
DATA DECODER	MLPX	076	4	12.09	Decoding 1 digit (4 to 16)
				14.15	Decoding 4 digits (4 to 16)
				24.01	Decoding 1 digit 8 to 256
				37.72	Decoding 2 digits (8 to 256)
DATA ENCODER	DMPX	077	4	11.90	Encoding 1 digit (16 to 4)
				58.70	Encoding 4 digits (16 to 4)
				19.76	Encoding 1 digit (256 to 8)
				80.32	Encoding 2 digits (256 to 8)
ASCII CONVERT	ASC	086	4	12.49	Converting 1 digit into ASCII
				18.03	Converting 4 digits into ASCII

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
ASCII TO HEX	HEX	162	4	12.64	Converting 1 digit
COLUMN TO LINE	LINE	063	4	34.95	---
LINE TO COLUMN	COLM	064	4	42.09	---
SIGNED BCD-TO-BINARY	BINS	470	4	15.73	Data format setting No. 0
				15.93	Data format setting No. 1
				15.93	Data format setting No. 2
				16.00	Data format setting No. 3
DOUBLE SIGNED BCD-TO-BINARY	BISL	472	4	18.59	Data format setting No. 0
				18.66	Data format setting No. 1
				18.41	Data format setting No. 2
				18.47	Data format setting No. 3
SIGNED BINARY-TO-BCD	BCDS	471	4	13.16	Data format setting No. 0
				13.18	Data format setting No. 1
				13.00	Data format setting No. 2
				13.12	Data format setting No. 3
DOUBLE SIGNED BINARY-TO-BCD	BDSL	473	4	13.74	Data format setting No. 0
				13.58	Data format setting No. 1
				13.79	Data format setting No. 2
				13.75	Data format setting No. 3
GRAY CODE CONVERSION	GRY	474	4	82.99	8-bit binary
				81.67	8-bit BCD
				98.65	8-bit angle
				97.67	15-bit binary
				98.99	15-bit BCD
				110.67	15-bit angle
				97.00	360° binary
				108.33	360° BCD
				113.00	360° angle

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Logic Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
LOGICAL AND	ANDW	034	4	0.30	---
DOUBLE LOGICAL AND	ANDL	610	4	0.60	---
LOGICAL OR	ORW	035	4	0.45	---
DOUBLE LOGICAL OR	ORWL	611	4	0.60	---
EXCLUSIVE OR	XORW	036	4	0.45	---
DOUBLE EXCLUSIVE OR	XORL	612	4	0.60	---
EXCLUSIVE NOR	XNRW	037	4	0.45	---
DOUBLE EXCLUSIVE NOR	XNRL	613	4	0.60	---
COMPLEMENT	COM	029	2	0.45	---
DOUBLE COMPLEMENT	COML	614	2	0.80	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Special Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
BINARY ROOT	ROTB	620	3	43.99	---
BCD SQUARE ROOT	ROOT	072	3	49.32	---
ARITHMETIC PROCESS	APR	069	4	13.96	Designating SIN and COS
				30.51	Designating line-segment approximation
FLOATING POINT DIVIDE	FDIV	079	4	222.90	---
BIT COUNTER	BCNT	067	4	29.70	Counting 1 word

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Floating-point Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
FLOATING TO 16-BIT	FIX	450	3	13.03	---
FLOATING TO 32-BIT	FIXL	451	3	12.03	---
16-BIT TO FLOATING	FLT	452	3	8.97	---
32-BIT TO FLOATING	FLTL	453	3	9.92	---
FLOATING-POINT ADD	+F	454	4	12.60	---
FLOATING-POINT SUBTRACT	–F	455	4	12.70	---
FLOATING-POINT DIVIDE	/F	457	4	13.40	---
FLOATING-POINT MULTIPLY	*F	456	4	12.67	---
DEGREES TO RADIANS	RAD	458	3	15.00	---
RADIANS TO DEGREES	DEG	459	3	17.97	---
SINE	SIN	460	3	37.10	---
COSINE	COS	461	3	41.97	---
TANGENT	TAN	462	3	30.86	---
ARC SINE	ASIN	463	3	65.14	---
ARC COSINE	ACOS	464	3	31.26	---
ARC TANGENT	ATAN	465	3	53.07	---
SQUARE ROOT	SQRT	466	3	20.73	---
EXPONENT	EXP	467	3	53.07	---
LOGARITHM	LOG	468	3	50.08	---
EXPONENTIAL POWER	PWR	840	4	185.77	---
Floating Symbol Comparison	LD, AND, OR +=F	329	3	11.01	---
	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			
FLOATING- POINT TO ASCII	FSTR	448	4	46.57	---
ASCII TO FLOATING-POINT	FVAL	449	3	25.37	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Double-precision Floating-point Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
DOUBLE SYMBOL COMPARISON	LD, AND, OR +=D	335	3	16.04	---
	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			
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	841	3	15.63	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	842	3	14.90	---
16-BIT BINARY TO DOUBLE FLOATING	DBL	843	3	12.29	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	844	3	14.13	---
DOUBLE FLOATING-POINT ADD	+D	845	4	17.89	---
DOUBLE FLOATING-POINT SUBTRACT	–D	846	4	17.96	---
DOUBLE FLOATING-POINT MULTIPLY	*D	847	4	17.96	---
DOUBLE FLOATING-POINT DIVIDE	/D	848	4	37.09	---
DOUBLE DEGREES TO RADIANS	RADD	849	3	32.07	---
DOUBLE RADIANS TO DEGREES	DEGD	850	3	33.76	---
DOUBLE SINE	SIND	851	3	66.97	---
DOUBLE COSINE	COSD	852	3	55.89	---
DOUBLE TANGENT	TAND	853	3	85.56	---
DOUBLE ARC SINE	ASIND	854	3	22.64	---
DOUBLE ARC COSINE	ACOSD	855	3	15.64	---
DOUBLE ARC TANGENT	ATAND	856	3	14.91	---
DOUBLE SQUARE ROOT	SQRTD	857	3	46.97	---
DOUBLE EXPONENT	EXPD	858	3	102.49	---
DOUBLE LOGARITHM	LOGD	859	3	19.03	---
DOUBLE EXPONENTIAL POWER	PWRD	860	4	182.83	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Table Data Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SET STACK	SSET	630	3	16.97	Designating 5 words in stack area
				700.67	Designating 1,000 words in stack area
PUSH ONTO STACK	PUSH	632	3	14.20	---
FIRST IN FIRST OUT	FIFO	633	3	11.50	Designating 5 words in stack area
				1.48 ms	Designating 1,000 words in stack area

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
LAST IN FIRST OUT	LIFO	634	3	16.94	---
DIMENSION RECORD TABLE	DIM	631	5	30.69	---
SET RECORD LOCATION	SETR	635	4	12.82	---
GET RECORD NUMBER	GETR	636	4	15.78	---
DATA SEARCH	SRCH	181	4	29.11	Searching for 1 word
				4.86 ms	Searching for 1,000 words
SWAP BYTES	SWAP	637	3	22.67	Swapping 1 word
				3.79 ms	Swapping 1,000 words
FIND MAXIMUM	MAX	182	4	34.17	Searching for 1 word
				4.46 ms	Searching for 1,000 words
FIND MINIMUM	MIN	183	4	34.97	Searching for 1 word
				4.74 ms	Searching for 1,000 words
SUM	SUM	184	4	46.63	Adding 1 word
				2.37 ms	Adding 1,000 words
FRAME CHECKSUM	FCS	180	4	33.17	For 1-word table length
				3.30 ms	For 1,000-word table length
STACK SIZE READ	SNUM	638	3	12.21	---
STACK DATA READ	SREAD	639	4	14.24	---
STACK DATA OVERWRITE	SWRIT	640	4	13.20	---
STACK DATA INSERT	SINS	641	4	17.78	---
				758.04	For 1,000-word table
STACK DATA DELETE	SDEL	642	4	19.83	---
				763.61	For 1,000-word table

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Data Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
PID CONTROL	PID	190	4	550.12	Initial execution
				546.43	Sampling
				152.87	Not sampling
LIMIT CONTROL	LMT	680	4	27.1	---
DEAD BAND CONTROL	BAND	681	4	27.23	---
DEAD ZONE CONTROL	ZONE	682	4	26.43	---
TIME-PROPORTIONAL OUTPUT	TPO	685	4	19.85	OFF execution time
				86.03	ON execution time with duty designation or displayed output limit
				95.27	ON execution time with manipulated variable designation and output limit enabled
SCALING	SCL	194	4	23.30	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SCALING 2	SCL2	486	4	20.93	---
SCALING 3	SCL3	487	4	24.37	---
AVERAGE	AVG	195	4	63.4	Average of an operation
				540.87	Average of 64 operations
PID CONTROL WITH AUTOTUNING	PIDAT	191	4	740.97	Initial execution
				611.30	Sampling
				197.97	Not sampling
				212.86	Initial execution of autotuning
				548.97	Autotuning when sampling

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Subroutine Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SUBROUTINE CALL	SBS	91	2	2.04	---
SUBROUTINE ENTRY	SBN	92	2	---	---
SUBROUTINE RETURN	RET	93	1	1.80	---
MACRO	MCRO	99	4	47.9	---
GLOBAL SUBROUTINE CALL	GSBN	751	2	---	---
GLOBAL SUBROUTINE ENTRY	GRET	752	1	2.04	---
GLOBAL SUBROUTINE RETURN	GSBS	750	2	1.80	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Interrupt Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SET INTERRUPT MASK	MSKS	690	3	51.90	Set
				63.09	Reset
READ INTERRUPT MASK	MSKR	692	3	19.99	Set
				43.67	Reset
CLEAR INTERRUPT	CLI	691	3	49.46	Set
				38.93	Reset
DISABLE INTERRUPTS	DI	693	1	14.83	---
ENABLE INTERRUPTS	EI	694	1	27.44	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

High-speed Counter and Pulse Output Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
MODE CONTROL	INI	880	4	80.39	Starting high-speed counter comparison
				47.99	Stopping high-speed counter comparison
				47.99	Changing pulse output PV
				48.01	Changing high-speed counter PV
				27.92	Changing PV of counter in interrupt input mode
				48.45	Stopping pulse output
				26.08	Stopping PWM(891) output
HIGH-SPEED COUNTER PV READ	PRV	881	4	80.39	Reading pulse output PV
				40.92	Reading high-speed counter PV
				28.63	Reading PV of counter in interrupt input mode
				39.20	Reading pulse output status
				66.43	Reading high-speed counter status
				34.63	Reading PWM(891) status
				145.52	Reading high-speed counter range comparison results
				47.48	Reading frequency of high-speed counter 0
COUNTER FREQUENCY CONVERT	PRV2	883	4	20.03	---
COMPARISON TABLE LOAD	CTBL	882	4	221.63	Registering target value table and starting comparison for 1 target value
				9.578 ms	Registering target value table and starting comparison for 48 target values
				262.37	Registering range table and starting comparison
				166.03	Only registering target value table for 1 target value
				9.557 ms	Only registering target value table for 48 target values
				241.70	Only registering range table
SPEED OUTPUT	SPED	885	4	89.24	Continuous mode
				94.47	Independent mode
SET PULSES	PULS	886	4	32.63	---
PULSE OUTPUT	PLS2	887	5	103.19	---
ACCELERATION CONTROL	ACC	888	4	111.26	Continuous mode
				121.73	Independent mode
ORIGIN SEARCH	ORG	889	3	112.93	Origin search
				98.65	Origin return
PULSE WITH VARIABLE DUTY FACTOR	PWM	891	4	30.26	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Step Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
STEP DEFINE	STEP	008	2	36.10	Step control bit ON
				18.77	Step control bit OFF
STEP START	SNXT	009	2	10.35	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

I/O Unit Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
I/O REFRESH	IORF	097	3	119.50	Refreshing 1 input word for CPM1A Unit
				122.17	Refreshing 1 output word for CPM1A Unit
				282.20	Refreshing 1 input word for CJ-series Special I/O Unit
				390.50	Refreshing 1 output word for CJ-series Special I/O Unit
				1.58 ms	Refreshing 10 input words for CPM1A Unit
				1.50 ms	Refreshing 10 output words for CPM1A Unit
				720.83	Refreshing 60 input words for CJ-series Special I/O Unit
				1.032 ms	Refreshing 60 output words for CJ-series Special I/O Unit
7-SEGMENT DECODER	SDEC	078	4	12.53	---
DIGITAL SWITCH INPUT	DSW	210	6	85.43	4 digits, data input value: 0
				80.43	4 digits, data input value: F
				82.11	8 digits, data input value: 0
				75.23	8 digits, data input value: F
TEN KEY INPUT	TKY	211	4	17.49	Data input value: 0
				18.69	Data input value: F
HEXADECIMAL KEY INPUT	HKY	212	5	72.77	Data input value: 0
				75.63	Data input value: F
MATRIX INPUT	MTR	213	5	71.55	Data input value: 0
				79.77	Data input value: F
7-SEGMENT DISPLAY OUTPUT	7SEG	214	5	88.23	4 digits
				86.97	8 digits
INTELLIGENT I/O READ	IORD	222	4	232.10	First execution
				237.10	When busy
				229.57	At end
INTELLIGENT I/O WRITE	IOWR	223	4	261.10	First execution
				259.10	When busy
				259.77	At end
CPU BUS I/O REFRESH	DLNK	226	4	425.69	Allocated 1 word

Note (1) When a double-length operand is used, add 1 to the value shown in the length column in the following table.

(2) The execution times of IORD(222) and IOWR(223) depend on the Special I/O Unit for which the instruction is executed.

Serial Communications Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
PROTOCOL MACRO	PMCR	260	5	152.83	Sending 0 words, receiving 0 words
				186.37	Sending 249 words, receiving 249 words
TRANSMIT	TXD	236	4	107.67	Sending 1 byte
				1.22 ms	Sending 256 bytes
RECEIVE	RXD	235	4	149.3	Storing 1 byte
				1.33 ms	Storing 256 bytes
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	256	4	145.64	Sending 1 byte
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	255	4	44.48	Storing 1 byte
CHANGE SERIAL PORT SETUP	STUP	237	3	479.3	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Network Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
NETWORK SEND	SEND	090	4	174.63	---
NETWORK RECEIVE	RECV	098	4	173.97	---
DELIVER COMMAND	CMND	490	4	195.97	---
EXPLICIT MESSAGE SEND	EXPLT	720	4	228.63	---
EXPLICIT GET ATTRIBUTE	EGATR	721	4	203.30	---
EXPLICIT SET ATTRIBUTE	ESATR	722	3	197.30	---
EXPLICIT WORD READ	ECHRD	723	4	188.63	---
EXPLICIT WORD WRITE	ECHWR	724	4	181.97	---
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	256	4	205.05	---
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	255	4	200.44	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Display Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
DISPLAY MESSAGE	MSG	046	3	17.16	Displaying message
				15.43	Deleting displayed message
7-SEGMENT LED WORD DATA DISPLAY	SCH	047	3	48.13	---
7-SEGMENT LED CONTROL	SCTRL	048	2	36.40	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Clock Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
CALENDAR ADD	CADD	730	4	212.90	---
CALENDAR SUBTRACT	CSUB	731	4	176.23	---
HOURS TO SECONDS	SEC	065	3	34.19	---
SECONDS TO HOURS	HMS	066	3	40.95	---
CLOCK ADJUSTMENT	DATE	735	2	134.67	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Debugging Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
TRACE MEMORY SAMPLING	TRSM	045	1	201.33	Sampling 1 bit and 0 words
				1.12 ms	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 following table.

Failure Diagnosis Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
FAILURE ALARM	FAL	006	3	23.24	Recording errors
				266.57	Deleting errors (in order of priority)
				817.17	Deleting errors (all errors)
				305.33	Deleting errors (individually)
SEVERE FAILURE ALARM	FALS	007	3	---	---
FAILURE POINT DETECTION	FPD	269	4	245.07	When executed
				258.2	First time
				317.73	When executed
				316.4	First time

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Other Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
SET CARRY	STC	040	1	0.15	---
CLEAR CARRY	CLC	041	1	0.15	---
EXTEND MAXIMUM CYCLE TIME	WDT	094	2	23.94	---
SAVE CONDITION FLAGS	CCS	282	1	14.97	---
LOAD CONDITION FLAGS	CCL	283	1	17.83	---
CONVERT ADDRESS FROM CV	FRMCV	284	3	31.03	---
CONVERT ADDRESS TO CV	TOCV	285	3	34.90	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Block Programming Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
BLOCK PROGRAM BEGIN	BPRG	096	2	26.59	---
BLOCK PROGRAM END	BEND	801	1	24.19	---
BLOCK PROGRAM PAUSE	BPPS	811	2	18.13	---
BLOCK PROGRAM RESTART	BPRS	812	2	9.29	---
CONDITIONAL BLOCK EXIT	(Execution condition) EXIT	806	1	23.33	EXIT condition satisfied
				9.33	EXIT condition not satisfied
CONDITIONAL BLOCK EXIT	EXIT (bit address)	806	2	26.78	EXIT condition satisfied
				11.47	EXIT condition not satisfied
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT (bit address)	806	2	26.74	EXIT condition satisfied
				11.41	EXIT condition not satisfied
Branching	IF (execution condition)	802	1	7.4	IF true
				13.5	IF false
Branching	IF (relay number)	802	2	11.55	IF true
				13.55	IF false
Branching (NOT)	IF NOT (relay number)	802	2	11.61	IF true
				13.61	IF false
Branching	ELSE	803	1	7.71	IF true
				13.55	IF false
Branching	IEND	804	1	13.58	IF true
				7.49	IF false
ONE CYCLE AND WAIT	WAIT (execution condition)	805	1	27.53	WAIT condition satisfied
				6.15	WAIT condition not satisfied
ONE CYCLE AND WAIT	WAIT (relay number)	805	2	28.78	WAIT condition satisfied
				9.82	WAIT condition not satisfied
ONE CYCLE AND WAIT (NOT)	WAIT NOT (relay number)	805	2	26.27	WAIT condition satisfied
				9.78	WAIT condition not satisfied
COUNTER WAIT	CNTW	814	4	36.57	First execution
				36.40	Normal execution
	CNTWX	818	4	43.69	First execution
				36.95	Normal execution
HIGH-SPEED TIMER WAIT	TMHW	815	3	48.37	First execution
				48.20	Normal execution
	TMHWX	817	3	50.59	First execution
				45.52	Normal execution
Loop Control	LOOP	809	1	17.03	---
Loop Control	LEND (execution condition)	810	1	17.13	LEND condition satisfied
				18.07	LEND condition not satisfied
Loop Control	LEND (relay number)	810	2	20.77	LEND condition satisfied
				23.63	LEND condition not satisfied
Loop Control	LEND NOT (relay number)	810	2	23.43	LEND condition satisfied
				20.97	LEND condition not satisfied

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
TIMER WAIT	TIMW	813	3	48.40	Default setting
				46.33	Normal execution
	TIMWX	816	3	48.02	Default setting
				47.09	Normal execution

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Text String Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
MOV STRING	MOV$	664	3	68.44	Transferring 1 character
CONCATENATE STRING	+$	656	4	145.10	1 character + 1 character
GET STRING LEFT	LEFT$	652	4	87.81	Retrieving 1 character from 2 characters
GET STRING RIGHT	RGHT$	653	4	91.81	Retrieving 1 character from 2 characters
GET STRING MIDDLE	MID$	654	5	94.77	Retrieving 1 character from 3 characters
FIND IN STRING	FIND$	660	4	82.81	Searching for 1 character from 2 characters
STRING LENGTH	LEN$	650	3	32.61	Detecting 1 character
REPLACE IN STRING	RPLC$	661	6	269.43	Replacing the first of 2 characters with 1 character
DELETE STRING	DEL$	658	5	114.00	Deleting the leading character of 2 characters
EXCHANGE STRING	XCHG$	665	3	108.54	Exchanging 1 character with 1 character
CLEAR STRING	CLR$	666	2	37.33	Clearing 1 character
INSERT INTO STRING	INS$	657	5	199.43	Inserting 1 character after the first of 2 characters
String Comparison Instructions	LD, AND, OR +=$	670	4	64.47	Comparing 1 character with 1 character
	LD, AND, OR +<>$	671			
	LD, AND, OR +<$	672			
	LD, AND, OR +>$	674			
	LD, AND, OR +>=$	675			

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Task Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
TASK ON	TKON	820	2	30.65	---
TASK OFF	TKOF	821	2	18.30	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Model Conversion Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
BLOCK TRANSFER	XFERC	565	4	37.04	Transferring 1 word
				2,922 ms	Transferring 1,000 words
SINGLE WORD DISTRIBUTE	DISTC	566	4	24.80	Data distribute
				35.57	Stack operation
DATA COLLECT	COLLC	567	4	29.83	Data distribute
				30.13	Stack operation
				31.10	Stack operation 1 word FIFO Read
				8,100 ms	Stack operation 1,000 word FIFO Read
MOVE BIT	MOVBC	568	4	28.03	---
BIT COUNTER	BCNTC	621	4	32.97	Counting 1 word
				5,703 ms	Counting 1,000 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

Special Function Block Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Conditions
GET VARIABLE ID	GETID	286	4	26.5	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2 Function Block Instance Execution Time

Use the following equation to calculate the effect of instance execution on the cycle time when function block definitions have been created and the instances copied into the user program.

Effect of Instance Execution on Cycle Time
= Startup time (A)
+ I/O parameter transfer processing time (B)
+ Execution time of instructions in function block definition (C)

The following table shows the length of time for A, B, and C.

Operation			CP1H CPU Unit
A	Startup time	Startup time not including I/O parameter transfer	6.8 μs
B	I/O parameter transfer processing time The data type is indicated in parentheses.	1-bit I/O variable (BOOL)	0.4 μs
		1-word I/O variable (INT, UINT, WORD)	0.3 μs
		2-word I/O variable (DINT, UDINT, DWORD, REAL)	0.5 μs
		4-word I/O variable (LINT, ULINT, LWORD, LREAL)	1.0 μs
C	Function block definition instruction execution time	Total instruction processing time (same as standard user program)	

Example:
Input variables with a 1-word data type (INT): 3
Output variables with a 1-word data type (INT): 2
Total instruction processing time in function block definition section: 10 μs
Execution time for 1 instance = 6.8 μs + (3 + 2) × 0.3 μs + 10 μs = 18.3 μs

Note The execution time is increased according to the number of multiple instances when the same function block definition has been copied to multiple locations.

Number of Function Block Program Steps

Use the following equation to calculate the number of program steps when function block definitions have been created and the instances copied into the user program.

Number of steps
= Number of instances × (Call part size m + I/O parameter transfer part size n × Number of parameters) + Number of instruction steps in the function block definition p (See note.)

Note The number of instruction steps in the function block definition (p) will not be diminished in subsequence instances when the same function block definition is copied to multiple locations (i.e., for multiple instances). Therefore, in the above equation, the number of instances is not multiplied by the number of instruction steps in the function block definition (p).

Contents			CP/CS/CJ-series CPU Units
m	Call part		57 steps
n	I/O parameter transfer part The data type is shown in parentheses.	1-bit I/O variable (BOOL)	6 steps
		1-word I/O variable (INT, UINT, WORD)	6 steps
		2-word I/O variable (DINT, UDINT, DWORD, REAL)	6 steps
		4-word I/O variable (LINT, ULINT, LWORD, LREAL)	12 steps
p	Number of instruction steps in function block definition	The total number of instruction steps (same as standard user program) + 27 steps.	

Example:
Input variables with a 1-word data type (INT): 5
Output variables with a 1-word data type (INT): 5
Function block definition section: 100 steps
Number of steps for 1 instance = 57 + (5 + 5) × 6 steps + 100 steps + 27 steps = 244 steps

Appendix A

Instruction Classifications by Function

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

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Basic instructions	**Input**	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	---	---
	Output	OUT	OUTPUT	OUT NOT	OUTPUT NOT	---	---
Sequence input instructions	**---**	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	**---**	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	MILH	MULTI-INTERLOCK DIFFERENTIATION HOLD
		MILR	MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILC	MULTI-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

Classification	Sub-class		Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Timer and counter instructions	**BCD (See note.)**	**Timer (with timer numbers)**	TIM	TIMER	TIMH	HIGH-SPEED TIMER	TMHH	ONE-MS TIMER
			TTIM	ACCUMULATIVE TIMER	---	---	---	---
		Timer (without timer numbers)	TIML	LONG TIMER	MTIM	MULTI-OUTPUT TIMER	---	---
		Counter (with counter numbers)	CNT	COUNTER	CNTR	REVERSIBLE TIMER	CNR	RESET TIMER/ COUNTER
	Binary (See note.)	**Timer (with timer numbers)**	TIMX	TIMER	TIMHX	HIGH-SPEED TIMER	TMHHX	ONE-MS TIMER
			TTIMX	ACCUMULATIVE TIMER	---	---	---	---
		Timer (without timer numbers)	TIMLX	LONG TIMER	MTIMX	MULTI-OUTPUT TIMER	---	---
		Counter (with counter numbers)	CNTX	COUNTER	CNTRX	REVERSIBLE TIMER	CNRX	RESET TIMER/ COUNTER
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)	LD, AND, OR + = DT, <> DT, < DT, <= DT, > DT, >= DT (See note 1.)	Time comparison	---	---
	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	---	---	---	---
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	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
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	NSFL	SHIFT N-BIT DATA LEFT	NSFR	SHIFT N-BIT DATA RIGHT	---	---
	Shift n-bit	NASL	SHIFT N-BITS LEFT	NSLL	DOUBLE SHIFT N-BITS LEFT	NASR	SHIFT N-BITS RIGHT
		NSRL	DOUBLE SHIFT N-BITS 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	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
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
		SIGN	16-BIT TO 32-BIT SIGNED BINARY	---	---	---	---
	Decoder/ encoder	MLPX	DATA DECODER	DMPX	DATA ENCODER	---	---
	ASCII/HEX convert	ASC	ASCII CONVERT	HEX	ASCII TO HEX	---	---
	Line/column convert	LINE	COLUMN TO LINE	COLM	LINE TO COLUMN	---	---
	Signed binary/BCD convert	BINS	SIGNED BCD-TO- BINARY	BISL	DOUBLE SIGNED BCD-TO- BINARY	BCDS	SIGNED BINARY-TO-BCD
		BDSL	DOUBLE SIGNED BINARY-TO-BCD	---	---	---	---
	Gray scale convert	GRY	GRAY CODE CONVERSION	---	---	---	---
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	COMPLEMENT	COML	DOUBLE COMPLEMENT	---	---
Special math instructions	**---**	ROTB	BINARY ROOT	ROOT	BCD SQUARE ROOT	APR	ARITHMETIC PROCESS
		FDIV	FLOATING POINT DIVIDE	BCNT	BIT COUNTER	---	---
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 TANGENT	---	---
	Floating- point math	SQRT	SQUARE ROOT	EXP	EXPONENT	LOG	LOGARITHM
		PWR	EXPONENTIAL POWER	---	---	---	---
	Symbol comparison and conversion	LD, AND, OR + =, <>, <, <=, >, >= + F	Symbol comparison (single-precision floating point)	FSTR	FLOATING-POINT TO ASCII	FVAL	ASCII TO 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	**Stack processing**	SSET	SET STACK	PUSH	PUSH ONTO STACK	LIFO	LAST IN FIRST OUT
		FIFO	FIRST IN FIRST OUT	SNUM	STACK SIZE READ	SREAD	STACK DATA READ
		SWRIT	STACK DATA OVERWRITE	SINS	STACK DATA INSERT	SDEL	STACK DATA DELETE
	1-record/multiple-word processing	DIM	DIMENSION RECORD TABLE	SETR	SET RECORD LOCATION	GETR	GET RECORD NUMBER
	Record-to-word processing	SRCH	DATA SEARCH	MAX	FIND MAXIMUM	MIN	FIND MINIMUM
		SUM	SUM	FCS	FRAME CHECKSUM	---	---
	Byte processing	SWAP	SWAP BYTES	---	---	---	---
Data control instructions	**---**	PID	PID CONTROL	PIDAT	PID CONTROL WITH AUTOTUNING	LMT	LIMIT CONTROL
		BAND	DEAD BAND CONTROL	ZONE	DEAD ZONE CONTROL	TPO	TIME-PROPORTIONAL OUTPUT
		SCL	SCALING	SCL2	SCALING 2	SCL3	SCALING 3
		AVG	AVERAGE	---	---	---	---
Subroutines instructions	**---**	SBS	SUBROUTINE CALL	MCRO	MACRO	SBN	SUBROUTINE ENTRY
		RET	SUBROUTINE RETURN	GSBS	GLOBAL SUBROUTINE CALL	GSBN	GLOBAL SUBROUTINE ENTRY
		GRET	GLOBAL SUBROUTINE RETURN	---	---	---	---
Interrupt control instructions	**---**	MSKS	SET INTERRUPT MASK	MSKR	READ INTERRUPT MASK	CLI	CLEAR INTERRUPT
		DI	DISABLE INTERRUPTS	EI	ENABLE INTERRUPTS	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
High-speed counter/ pulse output instructions (See note.)	**---**	INI	MODE CONTROL	PRV	HIGH-SPEED COUNTER PV READ	PRV2	COUNTER FREQUENCY CONVERT
		CTBL	COMPARISON TABLE LOAD	SPED	SPEED OUTPUT	PULS	SET PULSES
		PLS2	PULSE OUTPUT	ACC	ACCELERATION Control	ORG	ORIGIN SEARCH
		PWM	PULSE WITH VARIABLE DUTY FACTOR	---	---	---	---
Step instructions	**---**	STEP	STEP DEFINE	SNXT	STEP START	---	---
I/O Unit instructions	**---**	IORF	I/O REFRESH	SDEC	7-SEGMENT DECODER	DSW	DIGITAL SWITCH INPUT
		TKY	TEN KEY INPUT	HKY	HEXADECIMAL KEY INPUT	MTR	MATRIX INPUT
		7SEG	7-SEGMENT DISPLAY OUTPUT	IORD	INTELLIGENT I/O READ	IOWR	INTELLIGENT I/O WRITE
		DLNK (See note.)	CPU BUS UNIT I/O REFRESH	---	---	---	---
Serial communications instructions	**---**	PMCR	PROTOCOL MACRO	TXD	TRANSMIT	RXD	RECEIVE
		STUP	CHANGE SERIAL PORT SETUP	---	---	---	---
Network instructions	**---**	SEND	NETWORK SEND	RECV	NETWORK RECEIVE	CMND	DELIVER COMMAND
		EXPLT	SEND GENERAL EXPICIT	EGATR	EXPLICIT GET ATTRIBUTE	ESATR	EXPLICIT SET ATTRIBUTE
		ECHRD	EXPLICIT WORD READ	ECHWR	EXPLICIT WORD WRITE	---	---
Display instructions	**---**	MSG	DISPLAY MESSAGE	SCH	7-SEGMENT LED WORD DATA DISPLAY	SCTRL	7-SEGMENT LED CONTROL
Clock instructions	**---**	CADD	CALENDAR ADD	CSUB	CALENDAR SUBTRACT	SEC	HOURS TO SECONDS
		HMS	SECONDS TO HOURS	DATE	CLOCK ADJUSTMENT	---	---
Debugging instructions	**---**	TRSM	TRACE MEMORY SAMPLING	---	---	---	---
Failure diagnosis instructions	**---**	FAL	FAILURE ALARM	FALS	SEVERE FAILURE ALARM	FPD	FAILURE POINT DETECTION
Other instructions	**---**	STC	SET CARRY	CLC	CLEAR CARRY	---	---
		WDT	EXTEND MAXIMUM CYCLE TIME	CCS	SAVE CONDITION FLAGS	CCL	LOAD CONDITION FLAGS
		FRMCV	CONVERT ADDRESS FROM CV	TOCV	CONVERT ADDRESS TO CV	---	---

Classification	Sub-class		Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Block programming instructions	**Define block program area**		BPRG	BLOCK PROGRAM BEGIN	BEND	BLOCK PROGRAM END	---	---
	Block program start/stop		BPPS	BLOCK PROGRAM PAUSE	BPRS	BLOCK PROGRAM RESTART	---	---
	EXIT		EXIT *bit_address*	Conditional END	EXIT NOT *bit_address*	Conditional END NOT	*input_condition* EXIT	Conditional 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	---	---	---	---
	WAIT		WAIT *bit_address*	ONE CYCLE AND WAIT	WAIT NOT *bit_address*	ONE CYCLE AND WAIT NOT	*input_condition* WAIT	ONE CYCLE AND WAIT
	Timer/ counter	**BCD (See note.)**	TIMW	TIMER WAIT	CNTW	COUNTER WAIT	TMHW	HIGH-SPEED TIMER WAIT
		Binary (See note.)	TIMWX	TIMER WAIT	CNTWX	COUNTER WAIT	TMHWX	HIGH-SPEED TIMER WAIT
	Repeat		LOOP	LOOP BLOCK	LEND *bit_address*	LOOP BLOCK END	LEND NOT *bit_address*	LOOP BLOCK END NOT
			input_condition LEND	LOOP BLOCK END	---	---	---	---
Text string processing instructions	**---**		MOV$	MOV STRING	+$	CONCATENATE STRING	LEFT$	GET STRING LEFT
			RIGHT$	GET STRING RIGHT	MID$	GET STRING MIDDLE	FIND$	FIND IN STRING
			LEN$	STRING LENGTH	RPLC$	REPLACE IN STRING	DEL$	DELETE STRING
			XCHG$	EXCHANGE STRING	CLR$	CLEAR STRING	INS$	INSERT INTO STRING
			LD, AND, OR + =$, <>$, <$, <=$, >$, >=$	STRING COMPARISON	---	---	---	---
Task control instructions	**---**		TKON	TASK ON	TKOF	TASK OFF	---	---
Model conversion instruction	**---**		XFERC	BLOCK TRANSFER	DISTC	SINGLE WORD DISTRIBUTE	COLLC	DATA COLLECT
			MOVBC	MOVE BIT	BCNTC	BIT COUNTER	---	---
Special function block instructions	**---**		GETID	GET VARIABLE ID	---	---	---	---

Note Timers and counters are switched between BCD and binary data from the CX-Programmer.

Appendix B

List of Instructions by Function Code

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
---	LD	LOAD	@LD	%LD	!LD	89
---	LD NOT	LOAD NOT	---	---	!LD NOT	91
---	AND	AND	@AND	%AND	!AND	93
---	AND NOT	AND NOT	---	---	!AND NOT	95
---	OR	OR	@OR	%OR	!OR	97
---	OR NOT	OR NOT	---	---	!OR NOT	98
---	AND LD	AND LOAD	---	---	---	100
---	OR LD	OR LOAD	---	---	---	102
---	OUT	OUTPUT	---	---	!OUT	113
---	OUT NOT	OUTPUT NOT	---	---	!OUT NOT	114
---	SET	SET	@SET	%SET	!SET	122
---	RSET	RESET	@RSET	%RSET	!RSET	122
---	TIM	TIMER	---	---	---	171
---	TIMX	TIMER	---	---	---	171
---	CNT	COUNTER	---	---	---	194
000	NOP	NO OPERATION	---	---	---	133
001	END	END	---	---	---	132
002	IL	INTERLOCK	---	---	---	136
003	ILC	INTERLOCK CLEAR	---	---	---	136
004	JMP	JUMP	---	---	---	154
005	JME	JUMP END	---	---	---	154
006	FAL	FAILURE ALARM	@FAL	---	---	934
007	FALS	SEVERE FAILURE ALARM	---	---	---	942
008	STEP	STEP DEFINE	---	---	---	752
009	SNXT	STEP START	---	---	---	752
010	SFT	SHIFT REGISTER	---	---	---	275
011	KEEP	KEEP	---	---	!KEEP	115
012	CNTR	REVERSIBLE COUNTER	---	---	---	197
013	DIFU	DIFFERENTIATE UP	---	---	!DIFU	119
014	DIFD	DIFFERENTIATE DOWN	---	---	!DIFD	119
015	TIMH	HIGH-SPEED TIMER	---	---	---	175
016	WSFT	WORD SHIFT	@WSFT	---	---	282
017	ASFT	ASYNCHRONOUS SHIFT REGISTER	@ASFT	---	---	280
019	MCMP	MULTIPLE COMPARE	@MCMP	---	---	231
020	CMP	UNSIGNED COMPARE	---	---	!CMP	221
021	MOV	MOVE	@MOV	---	!MOV	248
022	MVN	MOVE NOT	@MVN	---	---	249
023	BIN	BCD-TO-BINARY	@BIN	---	---	390
024	BCD	BINARY-TO-BCD	@BCD	---	---	393
025	ASL	ARITHMETIC SHIFT LEFT	@ASL	---	---	284
026	ASR	ARITHMETIC SHIFT RIGHT	@ASR	---	---	287
027	ROL	ROTATE LEFT	@ROL	---	---	290
028	ROR	ROTATE RIGHT	@ROR	---	---	293

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
029	COM	COMPLEMENT	@COM	---	---	450
034	ANDW	LOGICAL AND	@ANDW	---	---	437
035	ORW	LOGICAL OR	@ORW	---	---	440
036	XORW	EXCLUSIVE OR	@XORW	---	---	443
037	XNRW	EXCLUSIVE NOR	@XNRW	---	---	446
040	STC	SET CARRY	@STC	---	---	958
041	CLC	CLEAR CARRY	@CLC	---	---	958
045	TRSM	TRACE MEMORY SAMPLING	---	---	---	930
046	MSG	DISPLAY MESSAGE	@MSG	---	---	909
047	SCH	7-SEGMENT LED WORD DATA DISPLAY	@SCH	---	---	911
048	SCTRL	7-SEGMENT LED CONTROL	@SCTRL	---	---	913
058	BINL	DOUBLE BCD-TO-DOUBLE BINARY	@BINL	---	---	391
059	BCDL	DOUBLE BINARY-TO-BCD	@BCDL	---	---	394
060	CMPL	DOUBLE UNSIGNED COMPARE	---	---	---	223
062	XFRB	MULTIPLE BIT TRANSFER	@XFRB	---	---	258
063	LINE	COLUMN TO LINE	@LINE	---	---	416
064	COLM	LINE TO COLUMN	@COLM	---	---	418
065	SEC	HOURS TO SECONDS	@SEC	---	---	922
066	HMS	SECONDS TO HOURS	@HMS	---	---	925
067	BCNT	BIT COUNTER	@BCNT	---	---	471
068	BCMP	UNSIGNED BLOCK COMPARE	@BCMP	---	---	236
069	APR	ARITHMETIC PROCESS	@APR	---	---	457
070	XFER	BLOCK TRANSFER	@XFER	---	---	261
071	BSET	BLOCK SET	@BSET	---	---	263
072	ROOT	BCD SQUARE ROOT	@ROOT	---	---	454
073	XCHG	DATA EXCHANGE	@XCHG	---	---	265
074	SLD	ONE DIGIT SHIFT LEFT	@SLD	---	---	303
075	SRD	ONE DIGIT SHIFT RIGHT	@SRD	---	---	304
076	MLPX	DATA DECODER	@MLPX	---	---	401
077	DMPX	DATA ENCODER	@DMPX	---	---	405
078	SDEC	7-SEGMENT DECODER	@SDEC	---	---	771
079	FDIV	FLOATING POINT DIVIDE	@FDIV	---	---	468
080	DIST	SINGLE WORD DISTRIBUTE	@DIST	---	---	268
081	COLL	DATA COLLECT	@COLL	---	---	270
082	MOVB	MOVE BIT	@MOVB	---	---	254
083	MOVD	MOVE DIGIT	@MOVD	---	---	256
084	SFTR	REVERSIBLE SHIFT REGISTER	@SFTR	---	---	277
085	TCMP	TABLE COMPARE	@TCMP	---	---	234
086	ASC	ASCII CONVERT	@ASC	---	---	409
087	TTIM	ACCUMULATIVE TIMER	---	---	---	182

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
088	ZCP	AREA RANGE COMPARE	---	---	---	243
090	SEND	NETWORK SEND	@SEND	---	---	863
091	SBS	SUBROUTINE CALL	@SBS	---	---	669
092	SBN	SUBROUTINE ENTRY	---	---	---	679
093	RET	SUBROUTINE RETURN	---	---	---	681
094	WDT	EXTEND MAXIMUM CYCLE TIME	@WDT	---	---	959
096	BPRG	BLOCK PROGRAM BEGIN	---	---	---	976
097	IORF	I/O REFRESH	@IORF	---	---	768
098	RECV	NETWORK RECEIVE	@RECV	---	---	869
099	MCRO	MACRO	@MCRO	---	---	675
114	CPS	SIGNED BINARY COMPARE	---	---	!CPS	226
115	CPSL	DOUBLE SIGNED BINARY COMPARE	---	---	---	228
116	ZCPL	DOUBLE AREA RANGE COMPARE	---	---	---	245
160	NEG	2'S COMPLEMENT	@NEG	---	---	396
161	NEGL	DOUBLE 2'S COMPLEMENT	@NEGL	---	---	398
162	HEX	ASCII TO HEX	@HEX	---	---	412
180	FCS	FRAME CHECKSUM	@FCS	---	---	598
181	SRCH	DATA SEARCH	@SRCH	---	---	585
182	MAX	FIND MAXIMUM	@MAX	---	---	589
183	MIN	FIND MINIMUM	@MIN	---	---	592
184	SUM	SUM	@SUM	---	---	595
190	PID	PID CONTROL	---	---	---	616
191	PIDAT	PID CONTROL WITH AUTOTUNING	---	---	---	628
194	SCL	SCALING	@SCL	---	---	653
195	AVG	AVERAGE	---	---	---	665
210	DSW	DIGITAL SWITCH INPUT	---	---	---	774
211	TKY	TEN KEY INPUT	@TKY	---	---	778
212	HKY	HEXADECIMAL KEY INPUT	---	---	---	781
213	MTR	MATRIX INPUT	---	---	---	785
214	7SEG	7-SEGMENT DISPLAY OUTPUT	---	---	---	789
222	IORD	INTELLIGENT I/O READ	@IORD	---	---	793
223	IOWR	INTELLIGENT I/O WRITE	@IOWR	---	---	796
226	DLNK	CPU BUS UNIT I/O REFRESH	@DLNK	---	---	799
235	RXD	RECEIVE	@RXD	---	---	819
236	TXD	TRANSMIT	@TXD	---	---	814
237	STUP	CHANGE SERIAL PORT SETUP	@STUP	---	---	840
255	RXDU	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	@RXDU	---	---	832
256	TXDU	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	@TXDU	---	---	824

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
260	PMCR	PROTOCOL MACRO	@PMCR	---	---	805
269	FPD	FAILURE POINT DETECTION	---	---	---	948
282	CCS	SAVE CONDITION FLAGS	@CCS	---	---	961
283	CCL	LOAD CONDITION FLAGS	@CCL	---	---	963
284	FRMCV	CONVERT ADDRESS FROM CV	@FRMCV	---	---	964
285	TOCV	CONVERT ADDRESS TO CV	@TOCV	---	---	968
286	GETID	GET VARIABLE ID	@GETID	---	---	1059
300	AND =	AND EQUAL	---	---	---	210
300	LD =	LOAD EQUAL	---	---	---	210
300	OR =	OR EQUAL	---	---	---	210
301	AND =L	AND DOUBLE EQUAL	---	---	---	210
301	LD =L	LOAD DOUBLE EQUAL	---	---	---	210
301	OR =L	OR DOUBLE EQUAL	---	---	---	210
302	AND =S	AND SIGNED EQUAL	---	---	---	210
302	LD =S	LOAD SIGNED EQUAL	---	---	---	210
302	OR =S	OR SIGNED EQUAL	---	---	---	210
303	AND =SL	AND DOUBLE SIGNED EQUAL	---	---	---	210
303	LD =SL	LOAD DOUBLE SIGNED EQUAL	---	---	---	210
303	OR =SL	OR DOUBLE SIGNED EQUAL	---	---	---	210
305	AND <>	AND NOT EQUAL	---	---	---	210
305	LD <>	LOAD NOT EQUAL	---	---	---	210
305	OR <>	OR NOT EQUAL	---	---	---	210
306	AND <>L	AND DOUBLE NOT EQUAL	---	---	---	210
306	LD <>L	LOAD DOUBLE NOT EQUAL	---	---	---	210
306	OR <>L	OR DOUBLE NOT EQUAL	---	---	---	210
307	AND <>S	AND SIGNED NOT EQUAL	---	---	---	210
307	LD <>S	LOAD SIGNED NOT EQUAL	---	---	---	210
307	OR <>S	OR SIGNED NOT EQUAL	---	---	---	210
308	AND <>SL	AND DOUBLE SIGNED NOT EQUAL	---	---	---	210
308	LD <>SL	LOAD DOUBLE SIGNED NOT EQUAL	---	---	---	210
308	OR <>SL	OR DOUBLE SIGNED NOT EQUAL	---	---	---	210
310	AND <	AND LESS THAN	---	---	---	210
310	LD <	LOAD LESS THAN	---	---	---	210
310	OR <	OR LESS THAN	---	---	---	210
311	AND <L	AND DOUBLE LESS THAN	---	---	---	210
311	LD <L	LOAD DOUBLE LESS THAN	---	---	---	210
311	OR <L	OR DOUBLE LESS THAN	---	---	---	210

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
312	AND <S	AND SIGNED LESS THAN	---	---	---	210
312	LD <S	LOAD SIGNED LESS THAN	---	---	---	210
312	OR <S	OR SIGNED LESS THAN	---	---	---	210
313	AND <SL	AND DOUBLE SIGNED LESS THAN	---	---	---	210
313	LD <SL	LOAD DOUBLE SIGNED LESS THAN	---	---	---	210
313	OR <SL	OR DOUBLE SIGNED LESS THAN	---	---	---	210
315	AND <=	AND LESS THAN OR EQUAL	---	---	---	210
315	LD <=	LOAD LESS THAN OR EQUAL	---	---	---	210
315	OR <=	OR LESS THAN OR EQUAL	---	---	---	210
316	AND <=L	AND DOUBLE LESS THAN OR EQUAL	---	---	---	210
316	LD <=L	LOAD DOUBLE LESS THAN OR EQUAL	---	---	---	210
316	OR <=L	OR DOUBLE LESS THAN OR EQUAL	---	---	---	210
317	AND <=S	AND SIGNED LESS THAN OR EQUAL	---	---	---	210
317	LD <=S	LOAD SIGNED LESS THAN OR EQUAL	---	---	---	210
317	OR <=S	OR SIGNED LESS THAN OR EQUAL	---	---	---	210
318	AND <=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	---	---	---	210
318	LD <=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	---	---	---	210
318	OR <=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	---	---	---	210
320	AND >	AND GREATER THAN	---	---	---	210
320	LD >	LOAD GREATER THAN	---	---	---	210
320	OR >	OR GREATER THAN	---	---	---	210
321	AND >L	AND DOUBLE GREATER THAN	---	---	---	210
321	LD >L	LOAD DOUBLE GREATER THAN	---	---	---	210
321	OR >L	OR DOUBLE GREATER THAN	---	---	---	210
322	AND >S	AND SIGNED GREATER THAN	---	---	---	210
322	LD >S	LOAD SIGNED GREATER THAN	---	---	---	210
322	OR >S	OR SIGNED GREATER THAN	---	---	---	210
323	AND >SL	AND DOUBLE SIGNED GREATER THAN	---	---	---	210
323	LD >SL	LOAD DOUBLE SIGNED GREATER THAN	---	---	---	210
323	OR >SL	OR DOUBLE SIGNED GREATER THAN	---	---	---	210

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
325	AND >=	AND GREATER THAN OR EQUAL	---	---	---	210
325	LD >=	LOAD GREATER THAN OR EQUAL	---	---	---	210
325	OR >=	OR GREATER THAN OR EQUAL	---	---	---	210
326	AND >=L	AND DOUBLE GREATER THAN OR EQUAL	---	---	---	210
326	LD >=L	LOAD DOUBLE GREATER THAN OR EQUAL	---	---	---	210
326	OR >=L	OR DOUBLE GREATER THAN OR EQUAL	---	---	---	210
327	AND >=S	AND SIGNED GREATER THAN OR EQUAL	---	---	---	210
327	LD >=S	LOAD SIGNED GREATER THAN OR EQUAL	---	---	---	210
327	OR >=S	OR SIGNED GREATER THAN OR EQUAL	---	---	---	210
328	AND >=SL	AND DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	---	210
328	LD >=SL	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	---	210
328	OR >=SL	OR DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	---	210
329	AND =F	AND FLOATING EQUAL	---	---	---	513
329	LD =F	LOAD FLOATING EQUAL	---	---	---	513
329	OR =F	OR FLOATING EQUAL	---	---	---	513
330	AND <>F	AND FLOATING NOT EQUAL	---	---	---	513
330	LD <>F	LOAD FLOATING NOT EQUAL	---	---	---	513
330	OR <>F	OR FLOATING NOT EQUAL	---	---	---	513
331	AND <F	AND FLOATING LESS THAN	---	---	---	513
331	LD <F	LOAD FLOATING LESS THAN	---	---	---	513
331	OR <F	OR FLOATING LESS THAN	---	---	---	513
332	AND <=F	AND FLOATING LESS THAN OR EQUAL	---	---	---	513
332	LD <=F	LOAD FLOATING LESS THAN OR EQUAL	---	---	---	513
332	OR <=F	OR FLOATING LESS THAN OR EQUAL	---	---	---	513
333	AND >F	AND FLOATING GREATER THAN	---	---	---	513
333	LD >F	LOAD FLOATING GREATER THAN	---	---	---	513
333	OR >F	OR FLOATING GREATER THAN	---	---	---	513

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
334	AND >=F	AND FLOATING GREATER THAN OR EQUAL	---	---	---	513
334	LD >=F	LOAD FLOATING GREATER THAN OR EQUAL	---	---	---	513
334	OR >=F	OR FLOATING GREATER THAN OR EQUAL	---	---	---	513
335	AND =D	AND DOUBLE FLOATING EQUAL	---	---	---	564
335	LD =D	LOAD DOUBLE FLOATING EQUAL	---	---	---	564
335	OR =D	OR DOUBLE FLOATING EQUAL	---	---	---	564
336	AND <>D	AND DOUBLE FLOATING NOT EQUAL	---	---	---	564
336	LD <>D	LOAD DOUBLE FLOATING NOT EQUAL	---	---	---	564
336	OR <>D	OR DOUBLE FLOATING NOT EQUAL	---	---	---	564
337	AND <D	AND DOUBLE FLOATING LESS THAN	---	---	---	564
337	LD <D	LOAD DOUBLE FLOATING LESS THAN	---	---	---	564
337	OR <D	OR DOUBLE FLOATING LESS THAN	---	---	---	564
338	AND <=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	---	---	---	564
338	LD <=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	---	---	---	564
338	OR <=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	---	---	---	564
339	AND >D	AND DOUBLE FLOATING GREATER THAN	---	---	---	564
339	LD >D	LOAD DOUBLE FLOATING GREATER THAN	---	---	---	564
339	OR >D	OR DOUBLE FLOATING GREATER THAN	---	---	---	564
340	AND >=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	---	564
340	LD >=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	---	564
340	OR >=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	---	564
341	AND = DT	AND TIME EQUAL	---	---	---	216
341	LD = DT	LOAD TIME EQUAL	---	---	---	216
341	OR = DT	OR TIME EQUAL	---	---	---	216
342	AND <> DT	AND TIME NOT EQUAL	---	---	---	216

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
342	LD <> DT	LOAD TIME NOT EQUAL	---	---	---	216
342	OR <> DT	OR TIME NOT EQUAL	---	---	---	216
343	AND < DT	AND TIME LESS THAN	---	---	---	216
343	LD < DT	LOAD TIME LESS THAN	---	---	---	216
343	OR < DT	OR TIME LESS THAN	---	---	---	216
344	AND <= DT	AND TIME LESS THAN OR EQUAL	---	---	---	216
344	LD <= DT	LD TIME LESS THAN OR EQUAL	---	---	---	216
344	OR <= DT	OR TIME LESS THAN OR EQUAL	---	---	---	216
345	AND > DT	AND TIME GREATER THAN	---	---	---	216
345	LD > DT	LOAD TIME GREATER THAN	---	---	---	216
345	OR > DT	OR TIME GREATER THAN	---	---	---	216
346	AND >= DT	AND TIME GREATER THAN OR EQUAL	---	---	---	216
346	LD >= DT	LOAD TIME GREATER THAN OR EQUAL	---	---	---	216
346	OR >= DT	OR TIME GREATER THAN OR EQUAL	---	---	---	216
350	AND TST	AND BIT TEST	---	---	---	110
350	LD TST	LOAD BIT TEST	---	---	---	110
350	OR TST	OR BIT TEST	---	---	---	110
351	AND TSTN	AND BIT TEST NOT	---	---	---	110
351	LD TSTN	LOAD BIT TEST NOT	---	---	---	110
351	OR TSTN	OR BIT TEST NOT	---	---	---	110
400	+	SIGNED BINARY ADD WITHOUT CARRY	@+	---	---	338
401	+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	@+L	---	---	340
402	+C	SIGNED BINARY ADD WITH CARRY	@+C	---	---	342
403	+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	@+CL	---	---	344
404	+B	BCD ADD WITHOUT CARRY	@+B	---	---	346
405	+BL	DOUBLE BCD ADD WITHOUT CARRY	@+BL	---	---	347
406	+BC	BCD ADD WITH CARRY	@+BC	---	---	349
407	+BCL	DOUBLE BCD ADD WITH CARRY	@+BCL	---	---	350
410	–	SIGNED BINARY SUBTRACT WITHOUT CARRY	@–	---	---	352
411	–L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	@–L	---	---	354
412	–C	SIGNED BINARY SUBTRACT WITH CARRY	@–C	---	---	358
413	–CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	@–CL	---	---	360

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
414	−B	BCD SUBTRACT WITHOUT CARRY	@−B	---	---	362
415	−BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	@−BL	---	---	364
416	−BC	BCD SUBTRACT WITH CARRY	@−BC	---	---	367
417	−BCL	DOUBLE BCD SUBTRACT WITH CARRY	@−BCL	---	---	368
420	*	SIGNED BINARY MULTIPLY	@*	---	---	370
421	*L	DOUBLE SIGNED BINARY MULTIPLY	@*L	---	---	372
422	*U	UNSIGNED BINARY MULTIPLY	@*U	---	---	373
423	*UL	DOUBLE UNSIGNED BINARY MULTIPLY	@*UL	---	---	375
424	*B	BCD MULTIPLY	@*B	---	---	376
425	*BL	DOUBLE BCD MULTIPLY	@*BL	---	---	378
430	/	SIGNED BINARY DIVIDE	@/	---	---	379
431	/L	DOUBLE SIGNED BINARY DIVIDE	@/L	---	---	381
432	/U	UNSIGNED BINARY DIVIDE	@/U	---	---	383
433	/UL	DOUBLE UNSIGNED BINARY DIVIDE	@/UL	---	---	385
434	/B	BCD DIVIDE	@/B	---	---	386
435	/BL	DOUBLE BCD DIVIDE	@/BL	---	---	388
448	FSTR	FLOATING POINT TO ASCII	@FSTR	---	---	517
449	FVAL	ASCII TO FLOATING POINT	@FVAL	---	---	522
450	FIX	FLOATING TO 16-BIT	@FIX	---	---	531
451	FIXL	FLOATING TO 32-BIT	@FIXL	---	---	533
452	FLT	16-BIT TO FLOATING	@FLT	---	---	534
453	FLTL	32-BIT TO FLOATING	@FLTL	---	---	535
454	+F	FLOATING-POINT ADD	@+F	---	---	485
455	−F	FLOATING-POINT SUBTRACT	@−F	---	---	487
456	*F	FLOATING-POINT MULTIPLY	@*F	---	---	489
457	/F	FLOATING-POINT DIVIDE	@/F	---	---	491
458	RAD	DEGREES TO RADIANS	@RAD	---	---	493
459	DEG	RADIANS-TO DEGREES	@DEG	---	---	494
460	SIN	SINE	@SIN	---	---	496
461	COS	COSINE	@COS	---	---	497
462	TAN	TANGENT	@TAN	---	---	499
463	ASIN	ARC SINE	@ASIN	---	---	501
464	ACOS	ARC COSINE	@ACOS	---	---	503
465	ATAN	ARC TANGENT	@ATAN	---	---	504
466	SQRT	SQUARE ROOT	@SQRT	---	---	506
467	EXP	EXPONENT	@EXP	---	---	508
468	LOG	LOGARITHM	@LOG	---	---	510

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
470	BINS	SIGNED BCD-TO-BINARY	@BINS	---	---	420
471	BCDS	SIGNED BINARY-TO-BCD	@BCDS	---	---	426
472	BISL	DOUBLE SIGNED BCD-TO-BINARY	@BISL	---	---	423
473	BDSL	DOUBLE SIGNED BINARY-TO-BCD	@BDSL	---	---	428
474	GRY	GRAY CODE CONVERSION	@GRY	---	---	431
486	SCL2	SCALING 2	@SCL2	---	---	657
487	SCL3	SCALING 3	@SCL3	---	---	661
490	CMND	DELIVER COMMAND	@CMND	---	---	875
498	MOVL	DOUBLE MOVE	@MOVL	---	---	251
499	MVNL	DOUBLE MOVE NOT	@MVNL	---	---	252
502	BCMP2	EXPANDED BLOCK COMPARE	@BCMP2	---	---	239
510	CJP	CONDITIONAL JUMP	---	---	---	158
511	CJPN	CONDITIONAL JUMP	---	---	---	158
512	FOR	FOR-NEXT LOOPS	---	---	---	164
513	NEXT	FOR-NEXT LOOPS	---	---	---	164
514	BREAK	BREAK LOOP	---	---	---	167
515	JMP0	MULTIPLE JUMP	---	---	---	162
516	JME0	MULTIPLE JUMP END	---	---	---	162
517	MILH	MULTI-INTERLOCK DIFFERENTIATION HOLD	---	---	---	140
518	MILR	MULTI-INTERLOCK DIFFERENTIATIONRELEASE	---	---	---	140
519	MILC	MULTI-INTERLOCK CLEAR	---	---	---	140
520	NOT	NOT	---	---	---	108
521	UP	CONDITION ON	---	---	---	109
522	DOWN	CONDITION OFF	---	---	---	109
530	SETA	MULTIPLE BIT SET	@SETA	---	---	124
531	RSTA	MULTIPLE BIT RESET	@RSTA	---	---	124
532	SETB	SINGLE BIT SET	@SETB	---	!SETB	127
533	RSTB	SINGLE BIT RESET	@RSTB	---	!RSTB	127
534	OUTB	SINGLE BIT OUTPUT	@OUTB	---	!OUTB	130
540	TMHH	ONE-MS TIMER	---	---	---	179
542	TIML	LONG TIMER	---	---	---	185
543	MTIM	MULTI-OUTPUT TIMER	---	---	---	188
545	CNR	RESET TIMER/ COUNTER	@CNR	---	---	201
546	CNTX	COUNTER	---	---	---	194
547	CNRX	RESET TIMER/ COUNTER	---	---	---	201
548	CNTRX	REVERSIBLE COUNTER	---	---	---	197
550	TIMX	TIMER	---	---	---	171
551	TIMHX	HIGH-SPEED TIMER	---	---	---	175
552	TMHHX	ONE-MS TIMER	---	---	---	179
553	TIMLX	LONG TIMER	---	---	---	185
554	MTIMX	MULTI-OUTPUT TIMER	---	---	---	188

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
555	TTIMX	ACCUMULATIVE TIMER	---	---	---	182
560	MOVR	MOVE TO REGISTER	@MOVR	---	---	271
561	MOVRW	MOVE TIMER/ COUNTER PV TO REGISTER	@MOVRW	---	---	273
562	XCGL	DOUBLE DATA EXCHANGE	@XCGL	---	---	266
565	XFERC	BLOCK TRANSFER	@XFERC	---	---	1046
566	DISTC	SINGLE WORD DISTRIBUTE	@DISTC	---	---	1048
567	COLLC	DATA COLLECT	@COLLC	---	---	1051
568	MOVBC	MOVE BIT	@MOVBC	---	---	1056
570	ASLL	DOUBLE SHIFT LEFT	@ASLL	---	---	285
571	ASRL	DOUBLE SHIFT RIGHT	@ASRL	---	---	288
572	ROLL	DOUBLE ROTATE LEFT	@ROLL	---	---	291
573	RORL	DOUBLE ROTATE RIGHT	@RORL	---	---	295
574	RLNC	ROTATE LEFT WITHOUT CARRY	@RLNC	---	---	296
575	RRNC	ROTATE RIGHT WITHOUT CARRY	@RRNC	---	---	300
576	RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	@RLNL	---	---	298
577	RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	@RRNL	---	---	301
578	NSFL	SHIFT N-BIT DATA LEFT	@NSFL	---	---	306
579	NSFR	SHIFT N-BIT DATA RIGHT	@NSFR	---	---	308
580	NASL	SHIFT N-BITS LEFT	@NASL	---	---	310
581	NASR	SHIFT N-BITS RIGHT	@NASR	---	---	315
582	NSLL	DOUBLE SHIFT N-BITS LEFT	@NSLL	---	---	312
583	NSRL	DOUBLE SHIFT N-BITS RIGHT	@NSRL	---	---	318
590	++	INCREMENT BINARY	@++	---	---	321
591	++L	DOUBLE INCREMENT BINARY	@++L	---	---	323
592	– –	DECREMENT BINARY	@– –	---	---	325
593	– –L	DOUBLE DECREMENT BINARY	@– –L	---	---	327
594	++B	INCREMENT BCD	@++B	---	---	329
595	++BL	DOUBLE INCREMENT BCD	@++BL	---	---	331
596	– –B	DECREMENT BCD	@– –B	---	---	333
597	– –BL	DOUBLE DECREMENT BCD	@– –BL	---	---	335
600	SIGN	16-BIT TO 32-BIT SIGNED BINARY	@SIGN	---	---	399
610	ANDL	DOUBLE LOGICAL AND	@ANDL	---	---	438
611	ORWL	DOUBLE LOGICAL OR	@ORWL	---	---	441
612	XORL	DOUBLE EXCLUSIVE OR	@XORL	---	---	445
613	XNRL	DOUBLE EXCLUSIVE NOR	@XNRL	---	---	448

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
614	COML	DOUBLE COMPLEMENT	@COML	---	---	451
620	ROTB	BINARY ROOT	@ROTB	---	---	452
621	BCNTC	BIT COUNTER	@BCNTC	---	---	1058
630	SSET	SET STACK	@SSET	---	---	568
631	DIM	DIMENSION RECORD TABLE	@DIM	---	---	579
632	PUSH	PUSH ONTO STACK	@PUSH	---	---	571
633	FIFO	FIRST IN FIRST OUT	@FIFO	---	---	574
634	LIFO	LAST IN FIRST OUT	@LIFO	---	---	576
635	SETR	SET RECORD LOCATION	@SETR	---	---	581
636	GETR	GET RECORD NUMBER	@GETR	---	---	583
637	SWAP	SWAP BYTES	@SWAP	---	---	587
638	SNUM	STACK SIZE READ	@SNUM	---	---	601
639	SREAD	STACK DATA READ	@SREAD	---	---	604
640	SWRIT	STACK DATA WRITE	@SWRIT	---	---	607
641	SINS	STACK DATA INSERT	@SINS	---	---	610
642	SDEL	STACK DATA DELETE	@SDEL	---	---	613
650	LEN$	STRING LENGTH	@LEN$	---	---	1019
652	LEFT$	GET STRING LEFT	@LEFT$	---	---	1010
653	RGHT$	GET STRING RIGHT	@RGHT$	---	---	1013
654	MID$	GET STRING MIDDLE	@MID$	---	---	1015
656	+$	CONCATENATE STRING	@+$	---	---	1008
657	INS$	INS$	@INS$	---	---	1029
658	DEL$	DELETE STRING	@DEL$	---	---	1023
660	FIND$	FIND IN STRING	@FIND$	---	---	1017
661	RPLC$	REPLACE IN STRING	@RPLC$	---	---	1021
664	MOV$	MOV STRING	@MOV$	---	---	1006
665	XCHG$	EXCHANGE STRING	@XCHG$	---	---	1026
666	CLR$	CLEAR STRING	@CLR$	---	---	1027
670	AND =$	AND STRING EQUALS	---	---	---	1032
670	LD =$	LOAD STRING EQUALS	---	---	---	1032
670	OR =$	OR STRING EQUALS	---	---	---	1032
671	AND <>$	AND STRING NOT EQUAL	---	---	---	1032
671	LD <>$	LOAD STRING NOT EQUAL	---	---	---	1032
671	OR <>$	OR STRING NOT EQUAL	---	---	---	1032
672	AND <$	AND STRING LESS THAN	---	---	---	1032
672	LD <$	LOAD STRING LESS THAN	---	---	---	1032
672	OR <$	OR STRING LESS THAN	---	---	---	1032
673	AND <=$	AND STRING LESS THAN OR EQUALS	---	---	---	1032
673	LD <=$	LOAD STRING LESS THAN OR EQUAL	---	---	---	1032
673	OR <=$	OR STRING LESS THAN OR EQUALS	---	---	---	1032
674	AND >$	AND STRING GREATER THAN	---	---	---	1032

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
674	LD >$	LOAD STRING GREATER THAN	---	---	---	1032
674	OR >$	OR STRING GREATER THAN	---	---	---	1032
675	AND >=$	AND STRING GREATER THAN OR EQUALS	---	---	---	1032
675	LD >=$	LOAD STRING GREATER THAN OR EQUALS	---	---	---	1032
675	OR >=$	OR STRING GREATER THAN OR EQUALS	---	---	---	1032
680	LMT	LIMIT CONTROL	@LMT	---	---	638
681	BAND	DEAD BAND CONTROL	@BAND	---	---	640
682	ZONE	DEAD ZONE CONTROL	@ZONE	---	---	643
685	TPO	TIME-PROPORTIONAL OUTPUT	---	---	---	645
690	MSKS	SET INTERRUPT MASK	@MSKS	---	---	693
691	CLI	CLEAR INTERRUPT	@CLI	---	---	700
692	MSKR	READ INTERRUPT MASK	@MSKR	---	---	697
693	DI	DISABLE INTERRUPTS	@DI	---	---	703
694	EI	ENABLE INTERRUPTS	---	---	---	704
720	EXPLT	EXPLICIT MESSAGE SEND	@EXPLT	---	---	882
721	EGATR	EXPLICIT GET ATTRIBUTE	@EGATR	---	---	889
722	ESATR	EXPLICIT SET ATTRIBUTE	@ESATR	---	---	896
723	ECHRD	EXPLICIT WORD READ	@ECHRD	---	---	901
724	ECHWR	EXPLICIT WORD CLEAR	@ECHWR	---	---	905
730	CADD	CALENDAR ADD	@CADD	---	---	916
731	CSUB	CALENDAR SUBTRACT	@CSUB	---	---	919
735	DATE	CLOCK ADJUSTMENT	@DATE	---	---	927
750	GSBS	GLOBAL SUBROUTINE CALL	@GSBS	---	---	682
751	GSBN	GLOBAL SUBROUTINE ENTRY	---	---	---	689
752	GRET	GLOBAL SUBROUTINE RETURN	---	---	---	692
801	BEND	BLOCK PROGRAM END	---	---	---	976
802	IF	CONDITIONAL BRANCHING BLOCK	---	---	---	981
802	IF	CONDITIONAL BRANCHING BLOCK	---	---	---	981
802	IF NOT	CONDITIONAL BRANCHING BLOCK NOT	---	---	---	981
803	ELSE	ELSE	---	---	---	981
804	IEND	IF END	---	---	---	981

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
805	WAIT	ONE CYCLE AND WAIT	---	---	---	988
805	WAIT	ONE CYCLE AND WAIT	---	---	---	988
805	WAIT NOT	ONE CYCLE AND WAIT NOT	---	---	---	988
806	EXIT	CONDITIONAL BLOCK EXIT	---	---	---	985
806	EXIT	CONDITIONAL BLOCK EXIT	---	---	---	985
806	EXIT NOT	CONDITIONAL BLOCK EXIT NOT	---	---	---	985
809	LOOP	LOOP	---	---	---	1001
810	LEND	LOOP END	---	---	---	1001
810	LEND	LOOP END	---	---	---	1001
810	LEND NOT	LOOP END NOT	---	---	---	1001
811	BPPS	BLOCK PROGRAM PAUSE	---	---	---	976
812	BPRS	BLOCK PROGRAM RESTART	---	---	---	979
813	TIMW	TIMER WAIT	---	---	---	992
814	CNTW	COUNTER WAIT	---	---	---	995
815	TMHW	HIGH-SPEED TIMER WAIT	---	---	---	998
816	TIMWX	TIMER WAIT	---	---	---	992
817	TMHWX	HIGH-SPEED TIMER WAIT	---	---	---	998
818	CNTWX	COUNTER WAIT	---	---	---	995
820	TKON	TASK ON	@TKON	---	---	1037
821	TKOF	TASK OFF	@TKOF	---	---	1040
840	PWR	EXPONENTIAL POWER	@PWR	---	---	512
841	FIXD	DOUBLE FLOATING TO 16-BIT BINARY	@FIXD	---	---	531
842	FIXLD	DOUBLE FLOATING TO 32-BIT BINARY	@FIXLD	---	---	533
843	DBL	16-BIT BINARY TO DOUBLE FLOATING	@DBL	---	---	534
844	DBLL	32-BIT BINARY TO DOUBLE FLOATING	@DBLL	---	---	535
845	+D	DOUBLE FLOATING-POINT ADD	@+D	---	---	537
846	–D	DOUBLE FLOATING-POINT SUBTRACT	@–D	---	---	539
847	*D	DOUBLE FLOATING-POINT MULTIPLY	@*D	---	---	541
848	/D	DOUBLE FLOATING-POINT DIVIDE	@/D	---	---	543
849	RADD	DOUBLE DEGREES TO RADIANS	@RADD	---	---	545
850	DEGD	DOUBLE RADIANS TO DEGREES	@RADD	---	---	546
851	SIND	DOUBLE SINE	@SIND	---	---	548
852	COSD	DOUBLE COSINE	@COSD	---	---	549
853	TAND	DOUBLE TANGENT	@TAND	---	---	551
854	ASIND	DOUBLE ARC SINE	@ASIND	---	---	552
855	ACOSD	DOUBLE ARC COSINE	@ACOSD	---	---	554
856	ATAND	DOUBLE ARC TANGENT	@ATAND	---	---	556

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
857	SQRTD	DOUBLE SQUARE ROOT	@SQRTD	---	---	558
858	EXPD	DOUBLE EXPONENT	@EXPD	---	---	559
859	LOGD	DOUBLE LOGARITHM	@LOGD	---	---	561
860	PWRD	DOUBLE EXPONENTIAL POWER	@PWRD	---	---	533
880	INI	MODE CONTROL	@INI	---	---	706
881	PRV	HIGH-SPEED COUNTER PV READ	@PRV	---	---	710
882	CTBL	COMPARISON TABLE LOAD	@CTBL	---	---	720
883	PRV2	COUNTER FREQUENCY CONVERT	@PRV2	---	---	716
885	SPED	SPEED OUTPUT	@SPED	---	---	724
886	PULS	SET PULSES	@PULS	---	---	729
887	PLS2	PULSE OUTPUT	@PLS2	---	---	731
888	ACC	ACCELERATION CONTROL	@ACC	---	---	739
889	ORG	ORIGIN SEARCH	@ORG	---	---	745
891	PWN	PULSE WITH VARIABLE DUTY FACTOR	@PWN	---	---	749

Appendix C
Alphabetical List of Instructions by Mnemonic

Symbols

Mnemonic	Function code	Instruction	Page
7SEG	214	7-SEGMENT DISPLAY OUTPUT	789
+	400	SIGNED BINARY ADD WITHOUT CARRY	338
+$	656	CONCATENATE STRING	1008
++	590	INCREMENT BINARY	321
++B	594	INCREMENT BCD	329
++BL	595	DOUBLE INCREMENT BCD	331
++L	591	DOUBLE INCREMENT BINARY	323
+B	404	BCD ADD WITHOUT CARRY	346
+BC	406	BCD ADD WITH CARRY	349
+BCL	407	DOUBLE BCD ADD WITH CARRY	350
+BL	405	DOUBLE BCD ADD WITHOUT CARRY	347
+C	402	SIGNED BINARY ADD WITH CARRY	342
+CL	403	DOUBLE SIGNED BINARY ADD WITH CARRY	344
+D	845	DOUBLE FLOATING-POINT ADD	537
+F	454	FLOATING-POINT ADD	485
+L	401	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	340
–	410	SIGNED BINARY SUBTRACT WITHOUT CARRY	352
– –	592	DECREMENT BINARY	325
– –B	596	DECREMENT BCD	333
– –BL	597	DOUBLE DECREMENT BCD	335
– –L	593	DOUBLE DECREMENT BINARY	327
–B	414	BCD SUBTRACT WITHOUT CARRY	362
–BC	416	BCD SUBTRACT WITH CARRY	367
–BCL	417	DOUBLE BCD SUBTRACT WITH CARRY	368
–BL	415	DOUBLE BCD SUBTRACT WITHOUT CARRY	364
–C	412	SIGNED BINARY SUBTRACT WITH CARRY	358
–CL	413	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	360
–D	846	DOUBLE FLOATING-POINT SUBTRACT	539
–F	455	FLOATING-POINT SUBTRACT	487
*	420	SIGNED BINARY MULTIPLY	370
*B	424	BCD MULTIPLY	376
*BL	425	DOUBLE BCD MULTIPLY	378
*D	847	DOUBLE FLOATING-POINT MULTIPLY	541
*F	456	FLOATING-POINT MULTIPLY	489
*L	421	DOUBLE SIGNED BINARY MULTIPLY	372
*U	422	UNSIGNED BINARY MULTIPLY	373
*UL	423	DOUBLE UNSIGNED BINARY MULTIPLY	375
–L	411	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	354
/	430	SIGNED BINARY DIVIDE	379
/B	434	BCD DIVIDE	386
/BL	435	DOUBLE BCD DIVIDE	388
/D	848	DOUBLE FLOATING-POINT DIVIDE	543

Mnemonic	Function code	Instruction	Page
/F	457	FLOATING-POINT DIVIDE	491
/L	431	DOUBLE SIGNED BINARY DIVIDE	381
/U	432	UNSIGNED BINARY DIVIDE	383
/UL	433	DOUBLE UNSIGNED BINARY DIVIDE	385

A

Mnemonic	Function code	Instruction	Page
ACC	888	ACCELERATION CONTROL	739
ACOS	464	ARC COSINE	503
ACOSD	855	DOUBLE ARC COSINE	556
AND	---	AND	93
AND <	310	AND LESS THAN	210
AND <$	672	AND STRING LESS THAN	1032
AND <>	305	AND NOT EQUAL	210
AND <>$	671	AND STRING NOT EQUAL	1032
AND <>D	336	AND DOUBLE FLOATING NOT EQUAL	564
AND <> DT	342	AND TIME NOT EQUAL	216
AND <>F	330	AND FLOATING NOT EQUAL	513
AND <>L	306	AND DOUBLE NOT EQUAL	210
AND <>S	307	AND SIGNED NOT EQUAL	210
AND <>SL	308	AND DOUBLE SIGNED NOT EQUAL	210
AND <D	337	AND DOUBLE FLOATING LESS THAN	564
AND <DT	343	AND TIME LESS THAN	216
AND <F	331	AND FLOATING LESS THAN	513
AND <L	311	AND DOUBLE LESS THAN	210
AND <S	312	AND SIGNED LESS THAN	210
AND <SL	313	AND DOUBLE SIGNED LESS THAN	210
AND =	300	AND EQUAL	210
AND =$	670	AND STRING EQUALS	1032
AND =D	335	AND DOUBLE FLOATING EQUAL	564
AND =DT	341	AND TIME EQUAL	216
AND =F	329	AND FLOATING EQUAL	513
AND =L	301	AND DOUBLE EQUAL	210
AND =S	302	AND SIGNED EQUAL	210
AND =SL	303	AND DOUBLE SIGNED EQUAL	210
AND >	320	AND GREATER THAN	210
AND >$	674	AND STRING GREATER THAN	1032
AND >D	339	AND DOUBLE FLOATING GREATER THAN	564
AND >DT	345	AND TIME GREATER THAN	216
AND >F	333	AND FLOATING GREATER THAN	513
AND >L	321	AND DOUBLE GREATER THAN	210
AND >S	322	AND SIGNED GREATER THAN	210
AND >SL	323	AND DOUBLE SIGNED GREATER THAN	210
AND LD	---	AND LOAD	100
AND NOT	---	AND NOT	95
AND TST	350	AND BIT TEST	110
AND TSTN	351	AND BIT TEST	110
AND <=	315	AND LESS THAN OR EQUAL	210
AND <=$	673	AND STRING LESS THAN OR EQUAL	1032

Mnemonic	Function code	Instruction	Page
AND <=D	338	AND DOUBLE FLOATING LESS THAN OR EQUAL	564
AND <=DT	344	AND TIME LESS THAN OR EQUAL	216
AND <=F	332	AND FLOATING LESS THAN OR EQUAL	513
AND <=L	316	AND DOUBLE LESS THAN OR EQUAL	210
AND <=S	317	AND SIGNED LESS THAN OR EQUAL	210
AND <=SL	318	AND DOUBLE SIGNED LESS THAN OR EQUAL	210
AND >=	325	AND GREATER THAN OR EQUAL	210
AND >=$	675	AND STRING GREATER THAN OR EQUALS	1032
AND >=D	340	AND DOUBLE FLOATING GREATER THAN OR EQUAL	564
AND >=DT	346	AND TIME GREATER THAN OR EQUAL	216
AND >=F	334	AND FLOATING GREATER THAN OR EQUAL	210
AND >=L	326	AND DOUBLE GREATER THAN OR EQUAL	210
AND >=S	327	AND SIGNED GREATER THAN OR EQUAL	210
AND >=SL	328	AND DOUBLE SIGNED GREATER THAN OR EQUAL	210
ANDL	610	DOUBLE LOGICAL AND	438
ANDW	034	LOGICAL AND	437
APR	069	ARITHMETIC PROCESS	457
ASC	086	ASCII CONVERT	409
ASFT	017	ASYNCHRONOUS SHIFT REGISTER	280
ASIN	463	ARC SINE	501
ASIND	854	DOUBLE ARC SINE	552
ASL	025	ARITHMETIC SHIFT LEFT	284
ASLL	570	DOUBLE SHIFT LEFT	285
ASR	026	ARITHMETIC SHIFT RIGHT	287
ASRL	571	DOUBLE SHIFT RIGHT	288
ATAN	465	ARC TANGENT	504
ATAND	856	DOUBLE ARC TANGENT	556
AVG	195	AVERAGE	665

B

Mnemonic	Function code	Instruction	Page
BAND	681	DEAD BAND CONTROL	640
BCD	024	BINARY-TO-BCD	393
BCDL	059	DOUBLE BINARY-TO-BCD	394
BCDS	471	SIGNED BINARY-TO-BCD	426
BCMP	068	UNSIGNED BLOCK COMPARE	236
BCMP2	502	EXPANDED BLOCK COMPARE	239
BCNT	067	BIT COUNTER	471
BCNTC	621	BIT COUNTER	1058
BDSL	473	DOUBLE SIGNED BINARY-TO-BCD	428
BEND	801	BLOCK PROGRAM END	976
BIN	023	BCD-TO-BINARY	390
BINL	058	DOUBLE BCD-TO-DOUBLE BINARY	391
BINS	470	SIGNED BCD-TO-BINARY	420
BISL	472	DOUBLE SIGNED BCD-TO-BINARY	423
BPPS	811	BLOCK PROGRAM PAUSE	976
BPRG	096	BLOCK PROGRAM BEGIN	976
BPRS	812	BLOCK PROGRAM RESTART	979

Mnemonic	Function code	Instruction	Page
BREAK	514	BREAK LOOP	167
BSET	071	BLOCK SET	263

C

Mnemonic	Function code	Instruction	Page
CADD	730	CALENDAR ADD	916
CCL	283	LOAD CONDITION FLAGS	963
CCS	282	SAVE CONDITION FLAGS	961
CJP	510	CONDITIONAL JUMP	158
CJPN	511	CONDITIONAL JUMP	158
CLC	041	CLEAR CARRY	958
CLI	691	CLEAR INTERRUPT	700
CLR$	666	CLEAR STRING	1027
CMND	490	DELIVER COMMAND	875
CMP	020	COMPARE	221
CMPL	060	DOUBLE COMPARE	223
CNR	545	RESET TIMER/COUNTER	201
CNRX	547	RESET TIMER/COUNTER	201
CNT	---	COUNTER	194
CNTX	546	COUNTER	194
CNTR	012	REVERSIBLE COUNTER	197
CNTRX	548	REVERSIBLE COUNTER	197
CNTW	814	COUNTER WAIT	995
CNTWX	818	COUNTER WAIT	995
COLL	081	DATA COLLECT	270
COLLC	567	DATA COLLECT	1051
COLM	064	LINE TO COLUMN	418
COM	029	COMPLEMENT	450
COML	614	DOUBLE COMPLEMENT	451
COS	461	COSINE	497
COSD	852	DOUBLE COSINE	549
CPS	114	SIGNED BINARY COMPARE	226
CPSL	115	DOUBLE SIGNED BINARY COMPARE	228
CSUB	731	CALENDAR SUBTRACT	919
CTBL	882	COMPARISON TABLE LOAD	720

D

Mnemonic	Function code	Instruction	Page
DATE	735	CLOCK ADJUSTMENT	927
DBL	843	16-BIT BINARY TO DOUBLE FLOATING	534
DBLL	844	32-BIT BINARY TO DOUBLE FLOATING	535
DEG	459	RADIANS-TO DEGREES	494
DEGD	850	DOUBLE RADIANS TO DEGREES	546
DEL$	658	DELETE STRING	1023
DI	693	DISABLE INTERRUPTS	703
DIFD	014	DIFFERENTIATE DOWN	119
DIFU	013	DIFFERENTIATE UP	119
DIM	631	DIMENSION RECORD TABLE	579
DIST	080	SINGLE WORD DISTRIBUTE	268

Mnemonic	Function code	Instruction	Page
DISTC	566	SINGLE WORD DISTRIBUTE	1048
DLNK	226	CPU BUS UNIT I/O REFRESH	799
DMPX	077	DATA ENCODER	405
DOWN	522	CONDITION OFF	109
DSW	210	DIGITAL SWITCH INPUT	774

E

Mnemonic	Function code	Instruction	Page
ECHRD	723	EXPLICIT WORD READ	901
ECHWR	724	EXPLICIT WORD WRITE	905
EGATR	721	EXPLICIT GET ATTRIBUTE	889
EI	694	ENABLE INTERRUPTS	704
ELSE	803	ELSE	981
END	001	END	132
ESATR	722	EXPLICIT SET ATTRIBUTE	896
EXIT NOT (operand)	806	CONDITIONAL BLOCK EXIT NOT	985
EXIT (input condition)	806	CONDITIONAL BLOCK EXIT	985
EXIT (operand)	806	CONDITIONAL BLOCK EXIT	985
EXP	467	EXPONENT	508
EXPD	858	DOUBLE EXPONENT	559
EXPLT	720	EXPLICIT MESSAGE SEND	882

F

Mnemonic	Function code	Instruction	Page
FAL	006	FAILURE ALARM	934
FALS	007	SEVERE FAILURE ALARM	942
FCS	180	FRAME CHECKSUM	598
FDIV	079	FLOATING POINT DIVIDE	468
FIFO	633	FIRST IN FIRST OUT	574
FIND$	660	FIND IN STRING	1017
FIX	450	FLOATING TO 16-BIT	531
FIXD	841	DOUBLE FLOATING TO 16-BIT BINARY	531
FIXL	451	FLOATING TO 32-BIT	533
FIXLD	842	DOUBLE FLOATING TO 32-BIT BINARY	533
FLT	452	16-BIT TO FLOATING	534
FLTL	453	32-BIT TO FLOATING	535
FOR	512	FOR-NEXT LOOPS	164
FPD	269	FAILURE POINT DETECTION	948
FRMCV	284	CONVERT ADDRESS FROM CV	964
FSTR	448	FLOATING POINT TO ASCII	517
FVAL	449	ASCII TO FLOATING POINT	522

G

Mnemonic	Function code	Instruction	Page
GETID	286	GET VARIABLE ID	1059
GETR	636	GET RECORD NUMBER	583
GRET	752	GLOBAL SUBROUTINE RETURN	692
GRY	474	GRAY CODE CONVERSION	431

Mnemonic	Function code	Instruction	Page
GSBN	751	GLOBAL SUBROUTINE ENTRY	689
GSBS	750	GLOBAL SUBROUTINE CALL	682

H

Mnemonic	Function code	Instruction	Page
HEX	162	ASCII TO HEX	412
HKY	212	HEXADECIMAL KEY INPUT	781
HMS	066	SECONDS TO HOURS	925

I

Mnemonic	Function code	Instruction	Page
IEND	804	IF END	981
IF NOT (operand)	802	IF NOT	981
IF (input condition)	802	IF	981
IF (operand)	802	IF	981
IL	002	INTERLOCK	136
ILC	003	INTERLOCK CLEAR	136
INI	880	MODE CONTROL	706
INS$	657	INS$	1029
IORD	222	INTELLIGENT I/O READ	793
IORF	097	I/O REFRESH	768
IOWR	223	INTELLIGENT I/O WRITE	796

J

Mnemonic	Function code	Instruction	Page
JME	005	JUMP END	154
JME0	516	MULTIPLE JUMP END	162
JMP	004	JUMP	154
JMP0	515	MULTIPLE JUMP	162

K

Mnemonic	Function code	Instruction	Page
KEEP	011	KEEP	115

L

Mnemonic	Function code	Instruction	Page
LD	---	LOAD	89
LD <	310	LOAD LESS THAN	210
LD <$	672	LOAD STRING LESS THAN	1032
LD <D	337	LOAD DOUBLE FLOATING LESS THAN	564
LD <DT	343	LOAD TIME LESS THAN	216
LD <F	331	LOAD FLOATING LESS THAN	513
LD <>	305	LOAD NOT EQUAL	210
LD <>$	671	LOAD STRING NOT EQUAL	1032
LD <>D	336	LOAD DOUBLE FLOATING NOT EQUAL	564
LD <>DT	342	LOAD TIME NOT EQUAL	216
LD <>F	330	LOAD FLOATING NOT EQUAL	513
LD <>L	306	LOAD DOUBLE NOT EQUAL	210

Mnemonic	Function code	Instruction	Page
LD <>S	307	LOAD SIGNED NOT EQUAL	210
LD <>SL	308	LOAD DOUBLE SIGNED NOT EQUAL	210
LD <L	311	LOAD DOUBLE LESS THAN	210
LD <S	312	LOAD SIGNED LESS THAN	210
LD <SL	313	LOAD DOUBLE SIGNED LESS THAN	210
LD =	300	LOAD EQUAL	210
LD =$	670	LOAD STRING EQUALS	1032
LD =D	335	LOAD DOUBLE FLOATING EQUAL	564
LD =DT	341	LOAD TIME EQUAL	216
LD =F	329	LOAD FLOATING EQUAL	513
LD =L	301	LOAD DOUBLE EQUAL	210
LD =S	302	LOAD SIGNED EQUAL	210
LD =SL	303	LOAD DOUBLE SIGNED EQUAL	210
LD >	320	LOAD GREATER THAN	210
LD >$	674	LOAD STRING GREATER THAN	1032
LD >D	339	LOAD DOUBLE FLOATING GREATER THAN	564
LD >DT	345	LOAD TIME GREATER THAN	216
LD >F	333	LOAD FLOATING GREATER THAN	513
LD >L	321	LOAD DOUBLE GREATER THAN	210
LD >S	322	LOAD SIGNED GREATER THAN	210
LD >SL	323	LOAD DOUBLE SIGNED GREATER THAN	210
LD NOT	---	LOAD NOT	91
LD TST	350	LOAD BIT TEST	110
LD TSTN	351	LOAD BIT TEST	110
LD <=	315	LOAD LESS THAN OR EQUAL	210
LD <=$	673	LOAD STRING LESS THAN OR EQUAL	1032
LD <=D	338	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	564
LD <=DT	344	LOAD TIME LESS THAN OR EQUAL	216
LD <=F	332	LOAD FLOATING LESS THAN OR EQUAL	513
LD <=L	316	LOAD DOUBLE LESS THAN OR EQUAL	210
LD <=S	317	LOAD SIGNED LESS THAN OR EQUAL	210
LD <=SL	318	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	210
LD >=	325	LOAD GREATER THAN OR EQUAL	210
LD >=$	675	LOAD STRING GREATER THAN OR EQUALS	1032
LD >=D	340	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	564
LD >=DT	346	LOAD TIME GREATER THAN OR EQUAL	216
LD >=F	334	LOAD FLOATING GREATER THAN OR EQUAL	513
LD >=L	326	LOAD DOUBLE GREATER THAN OR EQUAL	210
LD >=S	327	LOAD SIGNED GREATER THAN OR EQUAL	210
LD >=SL	328	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	210
LEFT$	652	GET STRING LEFT	1010
LEN$	650	STRING LENGTH	1019
LEND NOT (operand)	810	LOOP END NOT	1001
LEND (input condition)	810	LOOP END	1001
LEND (operand)	810	LOOP END	1001
LIFO	634	LAST IN FIRST OUT	576
LINE	063	COLUMN TO LINE	416

Mnemonic	Function code	Instruction	Page
LMT	680	LIMIT CONTROL	638
LOG	468	LOGARITHM	510
LOGD	859	DOUBLE LOGARITHM	561
LOOP	809	LOOP	1001

M

Mnemonic	Function code	Instruction	Page
MAX	182	FIND MAXIMUM	589
MCMP	019	MULTIPLE COMPARE	231
MCRO	099	MACRO	675
MID$	654	GET STRING MIDDLE	1015
MILC	519	MULTI-INTERLOCK CLEAR	140
MILH	517	MULTI-INTERLOCK DIFFERENTIATION HOLD	140
MILR	518	MULTI-INTERLOCK DIFFERENTIATION RELEASE	140
MIN	183	FIND MINIMUM	592
MLPX	076	DATA DECODER	401
MOV	021	MOVE	248
MOV$	664	MOVE STRING	1006
MOVB	082	MOVE BIT	254
MOVBC	568	MOVE BIT	1056
MOVD	083	MOVE DIGIT	256
MOVL	498	DOUBLE MOVE	251
MOVR	560	MOVE TO REGISTER	271
MOVRW	561	MOVE TIMER/COUNTER PV TO REGISTER	273
MSG	046	DISPLAY MESSAGE	909
MSKR	692	READ INTERRUPT MASK	697
MSKS	690	SET INTERRUPT MASK	693
MTIM	543	MULTI-OUTPUT TIMER	188
MTIMX	554	MULTI-OUTPUT TIMER	188
MTR	213	MATRIX INPUT	785
MVN	022	MOVE NOT	249
MVNL	499	DOUBLE MOVE NOT	252

N

Mnemonic	Function code	Instruction	Page
NASL	580	SHIFT N-BITS LEFT	310
NASR	581	SHIFT N-BITS RIGHT	315
NEG	160	2'S COMPLEMENT	396
NEGL	161	DOUBLE 2'S COMPLEMENT	398
NEXT	513	FOR-NEXT LOOPS	164
NOP	000	NO OPERATION	133
NOT	520	NOT	108
NSFL	578	SHIFT N-BIT DATA LEFT	306
NSFR	579	SHIFT N-BIT DATA RIGHT	308
NSLL	582	DOUBLE SHIFT N-BITS LEFT	312
NSRL	583	DOUBLE SHIFT N-BITS RIGHT	318

O

Mnemonic	Function code	Instruction	Page
OR	---	OR	97
OR <	310	OR LESS THAN	210
OR <$	672	OR STRING LESS THAN	1032
OR <>	305	OR NOT EQUAL	210
OR <>$	671	OR STRING NOT EQUAL	1032
OR <>D	336	OR DOUBLE FLOATING NOT EQUAL	564
OR <>DT	342	OR TIME NOT EQUAL	216
OR <>F	330	OR FLOATING NOT EQUAL	513
OR <>L	306	OR DOUBLE NOT EQUAL	210
OR <>S	307	OR SIGNED NOT EQUAL	210
OR <>SL	308	OR DOUBLE SIGNED NOT EQUAL	210
OR <D	337	OR DOUBLE FLOATING LESS THAN	564
OR <DT	343	OR TIME LESS THAN	216
OR <F	331	OR FLOATING LESS THAN	513
OR <L	311	OR DOUBLE LESS THAN	210
OR <S	312	OR SIGNED LESS THAN	210
OR <SL	313	OR DOUBLE SIGNED LESS THAN	210
OR =	300	OR EQUAL	210
OR =$	670	OR STRING EQUALS	1032
OR =D	335	OR DOUBLE FLOATING EQUAL	564
OR =DT	341	OR TIME EQUAL	216
OR =F	329	OR FLOATING EQUAL	513
OR =L	301	OR DOUBLE EQUAL	210
OR =S	302	OR SIGNED EQUAL	210
OR =SL	303	OR DOUBLE SIGNED EQUAL	210
OR >	320	OR GREATER THAN	210
OR >$	674	OR STRING GREATER THAN	1032
OR >D	339	OR DOUBLE FLOATING GREATER THAN	564
OR >DT	345	OR TIME GREATER THAN	216
OR >F	333	OR FLOATING GREATER THAN	513
OR >L	321	OR DOUBLE GREATER THAN	210
OR >S	322	OR SIGNED GREATER THAN	210
OR >SL	323	OR DOUBLE SIGNED GREATER THAN	210
OR LD	---	OR LOAD	102
OR NOT	---	OR NOT	98
OR TST	350	OR BIT TEST	110
OR TSTN	351	OR BIT TEST	110
OR <=	315	OR LESS THAN OR EQUAL	210
OR <=$	673	OR STRING LESS THAN OR EQUALS	1032
OR <=D	338	OR DOUBLE FLOATING LESS THAN OR EQUAL	564
OR <=DT	344	OR TIME LESS THAN OR EQUAL	216
OR <=F	332	OR FLOATING LESS THAN OR EQUAL	513
OR <=L	316	OR DOUBLE LESS THAN OR EQUAL	210
OR <=S	317	OR SIGNED LESS THAN OR EQUAL	210
OR <=SL	318	OR DOUBLE SIGNED LESS THAN OR EQUAL	210
OR >=	325	OR GREATER THAN OR EQUAL	210
OR >=$	675	OR STRING GREATER THAN OR EQUALS	1032

Mnemonic	Function code	Instruction	Page
OR >=D	340	OR DOUBLE FLOATING GREATER THAN OR EQUAL	564
OR >=DT	346	OR TIME GREATER THAN OR EQUAL	216
OR >=F	334	OR FLOATING GREATER THAN OR EQUAL	513
OR >=L	326	OR DOUBLE GREATER THAN OR EQUAL	210
OR >=S	327	OR SIGNED GREATER THAN OR EQUAL	210
OR >=SL	328	OR DOUBLE SIGNED GREATER THAN OR EQUAL	210
ORG	889	ORIGIN SEARCH	745
ORW	035	LOGICAL OR	440
ORWL	611	DOUBLE LOGICAL OR	441
OUT	---	OUTPUT	113
OUTB	534	SINGLE BIT OUTPUT	130
OUT NOT	---	OUTPUT NOT	114

P

Mnemonic	Function code	Instruction	Page
PID	190	PID CONTROL	616
PIDAT	191	PID CONTROL WITH AUTOTUNING	628
PMCR	260	PROTOCOL MACRO	805
PRV	881	HIGH-SPEED COUNTER PV READ	710
PRV2	883	COUNTER FREQUENCY CONVERT	716
PULS	886	SET PULSES	729
PLS2	887	PULSE OUTPUT	731
PUSH	632	PUSH ONTO STACK	571
PWM	891	PULSE WITH VARIABLE DUTY FACTOR	749
PWR	840	EXPONENTIAL POWER	512
PWRD	860	DOUBLE EXPONENTIAL POWER	533

R

Mnemonic	Function code	Instruction	Page
RAD	458	DEGREES TO RADIANS	493
RADD	849	DOUBLE DEGREES TO RADIANS	545
RECV	098	NETWORK RECEIVE	869
RET	093	SUBROUTINE RETURN	681
RGHT$	653	GET STRING RIGHT	1013
RLNC	574	ROTATE LEFT WITHOUT CARRY	296
RLNL	576	DOUBLE ROTATE LEFT WITHOUT CARRY	298
ROL	027	ROTATE LEFT	290
ROLL	572	DOUBLE ROTATE LEFT	291
ROOT	072	BCD SQUARE ROOT	454
ROR	028	ROTATE RIGHT	293
RORL	573	DOUBLE ROTATE RIGHT	295
ROTB	620	BINARY ROOT	452
RPLC$	661	REPLACE IN STRING	1021
RRNC	575	ROTATE RIGHT WITHOUT CARRY	300
RRNL	577	DOUBLE ROTATE RIGHT WITHOUT CARRY	301
RSET	---	RESET	122
RSTA	531	MULTIPLE BIT RESET	124
RSTB	533	SINGLE BIT RESET	127

Mnemonic	Function code	Instruction	Page
RXD	235	RECEIVE	819
RXDU	255	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	824

S

Mnemonic	Function code	Instruction	Page
SBN	092	SUBROUTINE ENTRY	679
SBS	091	SUBROUTINE CALL	669
SCH	047	7-SEGMENT LED CONTROL	911
SCL	194	SCALING	653
SCL2	486	SCALING 2	657
SCL3	487	SCALING 3	661
SCTRL	048	7-SEGMENT LED WORD DATA DISPLAY	913
SDEC	078	7-SEGMENT DECODER	771
SDEL	642	STACK DATA DELETE	613
SEC	065	HOURS TO SECONDS	922
SEND	090	NETWORK SEND	863
SET	---	SET	122
SETA	530	MULTIPLE BIT SET	124
SETB	532	SINGLE BIT SET	127
SETR	635	SET RECORD LOCATION	581
SFT	010	SHIFT REGISTER	275
SFTR	084	REVERSIBLE SHIFT REGISTER	277
SIGN	600	16-BIT TO 32-BIT SIGNED BINARY	399
SIN	460	SINE	496
SIND	851	DOUBLE SINE	548
SINS	641	STACK DATA INSERT	610
SLD	074	ONE DIGIT SHIFT LEFT	303
SNUM	638	STACK SIZE READ	601
SNXT	009	STEP START	752
SPED	885	SPEED OUTPUT	724
SQRT	466	SQUARE ROOT	506
SQRTD	857	DOUBLE SQUARE ROOT	558
SRCH	181	DATA SEARCH	585
SRD	075	ONE DIGIT SHIFT RIGHT	304
SREAD	639	STACK DATA READ	604
SSET	630	SET STACK	568
STC	040	SET CARRY	958
STEP	008	STEP DEFINE	752
STUP	237	CHANGE SERIAL PORT SETUP	840
SUM	184	SUM	595
SWAP	637	SWAP BYTES	587
SWRIT	640	STACK DATA WRITE	607

T

Mnemonic	Function code	Instruction	Page
TAN	462	TANGENT	499
TAND	853	DOUBLE TANGENT	551
TCMP	085	TABLE COMPARE	234
TIM	---	TIMER	171

Mnemonic	Function code	Instruction	Page
TIMH	015	HIGH-SPEED TIMER	175
TIMHX	551	HIGH-SPEED TIMER	175
TIML	542	LONG TIMER	185
TIMLX	553	LONG TIMER	185
TIMW	813	TIMER WAIT	992
TIMWX	816	TIMER WAIT	992
TIMX	---	TIMER	171
TKOF	821	TASK OFF	1040
TKON	820	TASK ON	1037
TKY	211	TEN KEY INPUT	778
TMHH	540	ONE-MS TIMER	179
TMHHX	552	ONE-MS TIMER	179
TMHW	815	HIGH-SPEED TIMER WAIT	998
TMHWX	817	HIGH-SPEED TIMER WAIT	998
TOCV	285	CONVERT ADDRESS TO CV	1059
TPO	685	TIME-PROPORTIONAL OUTPUT	645
TRSM	045	TRACE MEMORY SAMPLING	930
TTIM	087	ACCUMULATIVE TIMER	182
TTIMX	555	ACCUMULATIVE TIMER	182
TXD	236	TRANSMIT	814
TXDU	256	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	824

U

Mnemonic	Function code	Instruction	Page
UP	521	CONDITION ON	109

W

Mnemonic	Function code	Instruction	Page
WAIT NOT (operand)	805	ONE CYCLE AND WAIT NOT	988
WAIT (input condition)	805	ONE CYCLE AND WAIT	988
WAIT (operand)	805	ONE CYCLE AND WAIT	988
WDT	094	EXTEND MAXIMUM CYCLE TIME	959
WSFT	016	WORD SHIFT	282

X

Mnemonic	Function code	Instruction	Page
XCGL	562	DOUBLE DATA EXCHANGE	266
XCHG	073	DATA EXCHANGE	265
XCHG$	665	EXCHANGE STRING	1026
XFER	070	BLOCK TRANSFER	261
XFERC	565	BLOCK TRANSFER	1048
XFRB	062	MULTIPLE BIT TRANSFER	258
XNRL	613	DOUBLE EXCLUSIVE NOR	448
XNRW	037	EXCLUSIVE NOR	446
XORL	612	DOUBLE EXCLUSIVE OR	445
XORW	036	EXCLUSIVE OR	443

Z

Mnemonic	Function code	Instruction	Page
ZCP	088	AREA RANGE COMPARE	243
ZCPL	116	DOUBLE AREA RANGE COMPARE	245
ZONE	682	DEAD ZONE CONTROL	643

Index

A

B

C

G

H

I

J

L

M

N

O

P

R

S

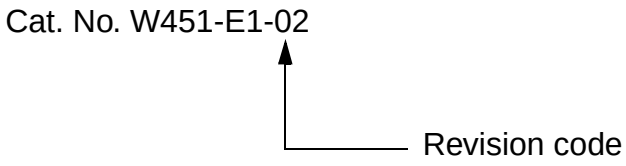
T

U

W

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	October 2005	Original production
02	May 2006	Information on CP1H CPU Units and Unit Versions added. **Pages xiii and 46 to 47:** CX-Programmer version updated. **Pages 694, 698, and 700:** Note corrected. **Pages 707 and 711:** Note added. **Page 714:** Table expanded. **Page 718:** *Range of Conversion Results* expanded and Overflow Flag added. **Pages 724, 732, and 739:** Frequency in illustration at bottom of page updated. **Page 732:** “S1” changed to “S” in illustrations. **Page 1080:** IOSP and IORS removed from table.

OMRON Corporation
Control Devices Division H.Q.
Shiokoji Horikawa, Shimogyo-ku,
Kyoto, 600-8530 Japan
Tel: (81)75-344-7109/Fax: (81)75-344-7149

Regional Headquarters

OMRON EUROPE B.V.
Wegalaan 67-69, NL-2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
1 East Commerce Drive, Schaumburg, IL 60173
U.S.A.
Tel: (1)847-843-7900/Fax: (1)847-843-8568

OMRON ASIA PACIFIC PTE. LTD.
83 Clemenceau Avenue,
#11-01, UE Square,
Singapore 239920
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

Authorized Distributor:

Cat. No. W451-E1-02 Note: Specifications subject to change without notice Printed in Japan