Cat. No. W474-E1-13

SYSMAC CS Series

CS1□-CPU□□□-□□

SYSMAC CJ Series

CJ2H-CPU6□-EIP
CJ2H-CPU6□
CJ2M-CPU□□
CJ1□-CPU□□□-□

SYSMAC One NSJ Series

NSJ□□-□□□□□-□□□

Programmable Controllers

INSTRUCTIONS REFERENCE MANUAL

OMRON

NOTE

Trademarks

Copyrights

SYSMAC CS Series

CS1□-CPU□□□-□□

SYSMAC CJ Series

CJ2H-CPU6□-EIP
CJ2H-CPU6□
CJ2M-CPU□□
CJ1□-CPU□□□-□

SYSMAC One NSJ Series

NSJ□□-□□□□□-□□□

Programmable Controllers

Instructions Reference Manual

About this Manual:

This manual describes the ladder diagram programming instructions of the CPU Units for CS/CJ-series Programmable Controllers (PLCs) and the NSJ-series Controllers. The CS Series, CJ Series and NSJ Series are subdivided as shown in the following table.

Series	Name	Model numbers
CJ Series	CJ2H CPU Units	CJ2H-CPU6□-EIP
		CJ2H-CPU6□
	CJ2M CPU Units	CJ2M-CPU□□
	CJ1-H CPU Units	CJ1H-CPU□□H-R
		CJ1□-CPU□□H
		CJ1□-CPU□□P
	CJ1M CPU Units	CJ1M-CPU□□
	CJ1 CPU Units	CJ1□-CPU□□
CS Series	CS1-H CPU Units	CS1G/H-CPU□□H
	CS1 CPU Units	CS1G/H-CPU□-EV1
	CS1D CPU Units	CS1D-CPU□□HA
		CS1D-CPU□□SA
		CS1D-CPU□□H
		CS1D-CPU□□S
		CS1D-CPU□□P
NSJ Series	NSJ-series Controllers	NSJ5-TQ□□(B)-G5D
		NSJ5-SQ□□(B)-G5D
		NSJ8-TV□□(B)-G5D
		NSJ10-TV□□(B)-G5D
		NSJ12-TS□□(B)-G5D
		NSJ5-TQ□□(B)-M3D
		NSJ5-SQ□□(B)-M3D
		NSJ8-TV□□(B)-M3D

NSJ-series Controller Notation

For information in this manual on the Controller Section of NSJ-series Controllers, refer to the information of the equivalent CJ-series PLC. The following models are equivalent.

NSJ-series Controllers	Equivalent CJ-series CPU Unit
NSJ□-TQ□□(B)-G5D	CJ1G-CPU45H CPU Unit with unit version 3.0
NSJ□-TQ□□(B)-M3D	CJ1G-CPU45H CPU Unit with unit version 3.0 (See note.)

Note: The following points differ between the NSJ□-TQ□□(B)-M3D and the CJ1G-CPU45H.

Item		CJ-series CPU Unit CJ1G-CPU45H	Controller Section in NSJ□-□□□□(B)-M3D
I/O capacity		1280 points	640 points
Program capacity		60 Ksteps	20 Ksteps
No. of Expansion Racks		3 max.	1 max.
EM Area		32 Kwords x 3 banks E0_00000 to E2_32767	None
Function blocks	Max. No. of definitions	1024	128
	Max. No. of instances	2048	256
Capacity in built-in file memory	FB program memory	1024 KB	256 KB
	Variable tables	128 KB	64K KB

Please read this manual and all related manuals listed in the table on the next page and be sure you understand information provided before attempting to program or use CS/CJ-series CPU Units in a PLC System.

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 Programming Device displays to mean Programmable Controller.

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.

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- **Exclusive Warranty**

  Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.

- **Limitations**

  OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.

  Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

- **Buyer Remedy**

  Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses 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. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

See http://www.omron.com/global/ or contact your Omron representative for published information.

Limitation on Liability; Etc

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

Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.

NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY OR IN LARGE QUANTITIES WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials 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 user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

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 part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

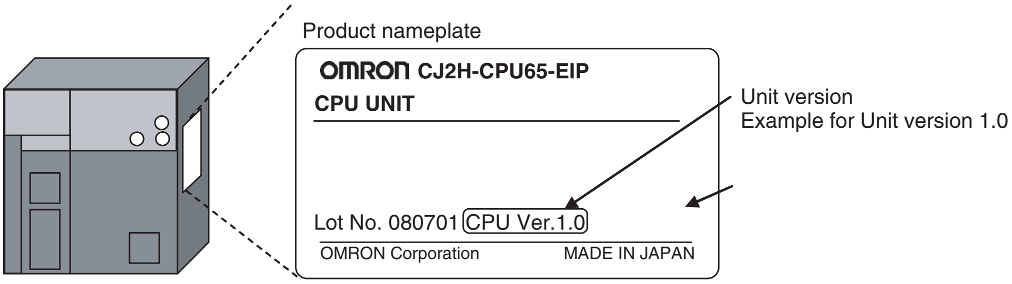
Unit Versions of CS/CJ-series CPU Units

Unit Versions

A “unit version” has been introduced to manage CPU Units in the CS/CJ Series according to differences in functionality accompanying Unit upgrades. Some instructions are supported only by specific versions of a CPU Unit. Confirm support before programming.

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

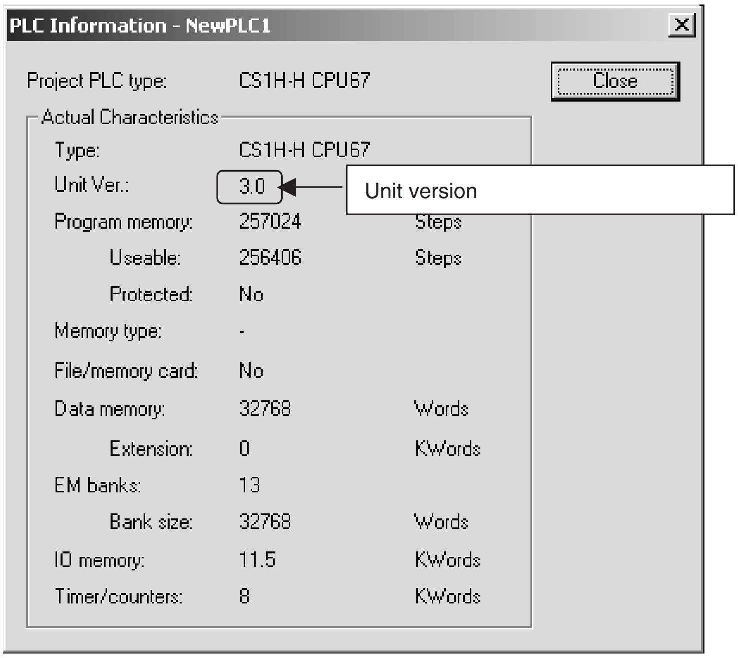
CX-Programmer version 4.0 can be used to confirm the unit version using one of the following two methods.

- Using the ***PLC Information***
- Using the ***Unit Manufacturing Information*** (This method can be used for Special I/O Units and CPU Bus Units as well.)

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.



Project PLC type: CS1H-H CPU67

Actual Characteristics

Type: CS1H-H CPU67

Unit Ver.: 3.0

Program memory: 257024 Steps

Useable: 256406 Steps

Protected: No

Memory type: -

File/memory card: No

Data memory: 32768 Words

Extension: 0 KWords

EM banks: 13

Bank size: 32768 Words

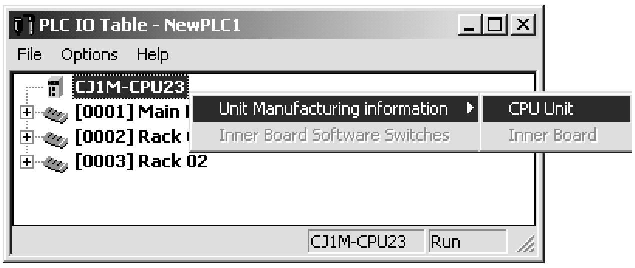
IO memory: 11.5 KWords

Timer/counters: 8 KWords

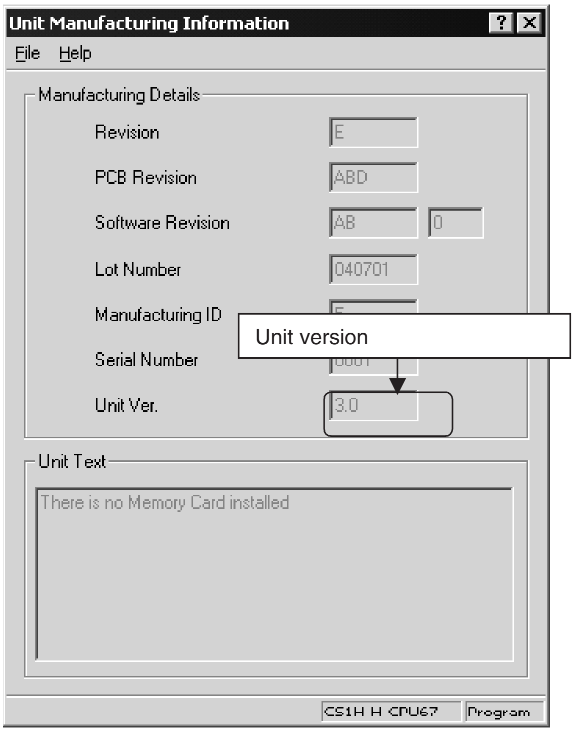
Use the above display to confirm the unit version of the CPU Unit.

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.



Manufacturing Details

Revision: E

PCB Revision: ABD

Software Revision: AB 0

Lot Number: 040701

Manufacturing ID: E

Serial Number: 0001

Unit Ver.: 3.0

Unit Text

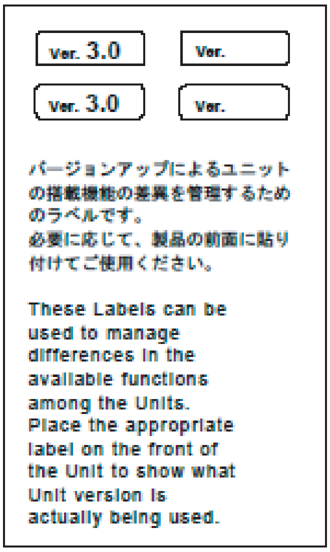
There is no Memory Card installed

CS1H H CPU67 Program

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.



Ver. 3.0 Ver.

Ver. 3.0 Ver.

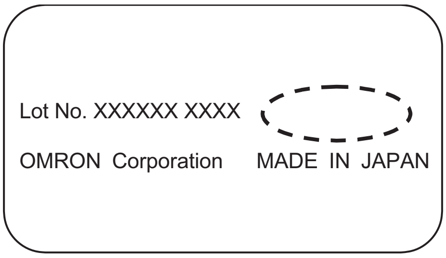
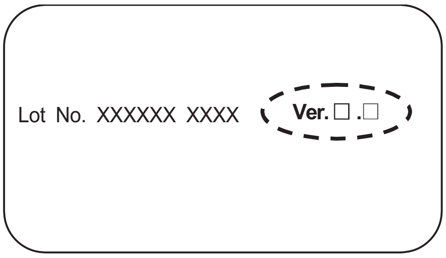
バージョンアップによるユニットの搭載機能の差異を管理するためのラベルです。
必要に応じて、製品の前面に貼り付けてご使用ください。

These Labels can be used to manage differences in the available functions among the Units.
Place the appropriate label on the front of the Unit to show what Unit version is actually being used.

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

Unit Version Notation

In this manual, the unit version of a CPU Unit is given as shown in the following table.

Product nameplate **Meaning**	**CPU Units on which no unit version is given** 	**Units on which a version is given (Ver. □.□)** 
Designating individual CPU Units (e.g., the CS1H-CPU67H)	Pre-Ver. 2.0 CS1-H CPU Units	CS1H-CPU67H CPU Unit Ver. □.□
Designating groups of CPU Units (e.g., the CS1-H CPU Units)	Pre-Ver. 2.0 CS1-H CPU Units	CS1-H CPU Units Ver. □.□
Designating an entire series of CPU Units (e.g., the CS-series CPU Units)	Pre-Ver. 2.0 CS-series CPU Units	CS-series CPU Units Ver. □.□

Unit Versions

CS Series

Units	Models	Unit version
CS1-H CPU Units	CS1□-CPU□□H	Unit version 4.0
		Unit version 3.0
		Unit version 2.0
		Pre-Ver. 2.0
CS1D CPU Units	Duplex-CPU Systems CS1D-CPU□□H	Unit version 1.2
		Unit version 1.1
		Pre-Ver. 1.1
	Single-CPU Systems CS1D-CPU□□S	Unit version 2.0
CS1 CPU Units	CS1□-CPU□□	No unit version.
CS1 Version-1 CPU Units	CS1□-CPU□□-V1	No unit version.

CJ Series

Units	Models	Unit version
CJ2 CPU Units	CJ2H-CPU□□-□□□	Unit version 1.3
		Unit version 1.2
		Unit version 1.1
		Unit version 1.0
	CJ2M-CPU□□	Unit version 1.0
CJ1-H CPU Units	CJ1H-CPU□□H-R	Unit version 4.0
	CJ1□-CPU□□H CJ1□-CPU□□P	Unit version 4.0
		Unit version 3.0
		Unit version 2.0
		Pre-Ver. 2.0
CJ1M CPU Units	CJ1M-CPU12/13 CJ1M-CPU22/23	Unit version 4.0
		Unit version 3.0
		Unit version 2.0
		Pre-Ver. 2.0
	CJ1M-CPU11/21	Unit version 4.0
		Unit version 3.0
		Unit version 2.0

NSJ Series

Units	Unit version
NSJ□-TQ□□(B)-G5D NSJ□-TQ□□(B)-M3D	Unit version 3.0

Function Support by Unit Version

• Functions Supported for Unit Version 4.0 or Later

CX-Programmer 7.0 or higher must be used to enable using the functions added for unit version 4.0.

CS1-H CPU Units

Function		CS1□-CPU□□H	
		Unit version 4.0 or later	Other unit versions
Online editing of function blocks **Note** This function cannot be used for simulations on the CX-Simulator.		OK	---
Input-output variables in function blocks		OK	---
Text strings in function blocks		OK	---
New application instructions	Number-Text String Conversion Instructions: NUM4, NUM8, NUM16, STR4, STR8, and STR16	OK	---
	TEXT FILE WRITE (TWRIT)	OK	---

CS1D CPU Units

Unit version 4.0 is not supported.

CJ1-H/CJ1M CPU Units

Function		CJ1H-CPU□□H-R, CJ1□-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□	
		Unit version 4.0 or later	Other unit versions
Online editing of function blocks **Note** This function cannot be used for simulations on the CX-Simulator.		OK	---
Input-output variables in function blocks		OK	---
Text strings in function blocks		OK	---
New application instructions	Number-Text String Conversion Instructions: NUM4, NUM8, NUM16, STR4, STR8, and STR16	OK	---
	TEXT FILE WRITE (TWRIT)	OK	---

User programs that contain functions supported only by CPU Units with unit version 4.0 or later cannot be used on CS/CJ-series CPU Units with unit version 3.0 or earlier. An error message will be displayed if an attempt is made to download programs containing unit version 4.0 functions to a CPU Unit with a unit version of 3.0 or earlier, and the download will not be possible.

If an object program file (.OBJ) using these functions is transferred to a CPU Unit with a unit version of 3.0 or earlier, a program error will occur when operation is started or when the unit version 4.0 function is executed, and CPU Unit operation will stop.

- **Functions Supported for Unit Version 3.0 or Later**

CX-Programmer 5.0 or higher must be used to enable using the functions added for unit version 3.0.

CS1-H CPU Units

Function		CS1□-CPU□□H	
		Unit version 3.0 or later	Other unit versions
Function blocks		OK	---
Serial Gateway (converting FINS commands to CompoWay/F commands at the built-in serial port)		OK	---
Comment memory (in internal flash memory)		OK	---
Expanded simple backup data		OK	---
New application instructions	TXDU(256), RXDU(255) (support no-protocol communications with Serial Communications Units with unit version 1.2 or later)	OK	---
	Model conversion instructions: XFERC(565), DISTC(566), COLLC(567), MOVBC(568), BCNTC(621)	OK	---
	Special function block instructions: GETID(286)	OK	---
Additional instruction functions	TXD(235) and RXD(236) instructions (support no-protocol communications with Serial Communications Boards with unit version 1.2 or later)	OK	---

CS1D CPU Units

Unit version 3.0 is not supported.

CJ1-H/CJ1M CPU Units

Function		CJ1H-CPU□□H-R, CJ1□-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□	
		Unit version 3.0 or later	Other unit versions
Function blocks		OK	---
Serial Gateway (converting FINS commands to CompoWay/F commands at the built-in serial port)		OK	---
Comment memory (in internal flash memory)		OK	---
Expanded simple backup data		OK	---
New application instructions	TXDU(256), RXDU(255) (support no-protocol communications with Serial Communications Units with unit version 1.2 or later)	OK	---
	Model conversion instructions: XFERC(565), DISTC(566), COLLC(567), MOVBC(568), BCNTC(621)	OK	---
	Special function block instructions: GETID(286)	OK	---
Additional instruction functions	PRV(881) and PRV2(883) instructions: Added high-frequency calculation methods for calculating pulse frequency. (CJ1M CPU Units only)	OK	---

User programs that contain functions supported only by CPU Units with unit version 3.0 or later cannot be used on CS/CJ-series CPU Units with unit version 2.0 or earlier. An error message will be displayed if an attempt is made to download programs containing unit version 3.0 functions to a CPU Unit with a unit version of 2.0 or earlier, and the download will not be possible.

If an object program file (.OBJ) using these functions is transferred to a CPU Unit with a unit version of 2.0 or earlier, a program error will occur when operation is started or when the unit version 3.0 function is executed, and CPU Unit operation will stop.

- **Functions Supported for Unit Version 2.0 or Later**

CX-Programmer 4.0 or higher must be used to enable using the functions added for unit version 2.0.

CS1-H CPU Units

Function		CS1-H CPU Units (CS1□-CPU□□H)	
		Unit version 2.0 or later	Other unit versions
Downloading and Uploading Individual Tasks		OK	---
Improved Read Protection Using Passwords		OK	---
Write Protection from FINS Commands Sent to CPU Units via Networks		OK	---
Online Network Connections without I/O Tables		OK	---
Communications through a Maximum of 8 Network Levels		OK	---
Connecting Online to PLCs via NS-series PTs		OK	OK from lot number 030201
Setting First Slot Words		OK for up to 64 groups	OK for up to 8 groups
Automatic Transfers at Power ON without a Parameter File		OK	---
Automatic Detection of I/O Allocation Method for Automatic Transfer at Power ON		---	---
Operation Start/End Times		OK	---
New Application Instructions	MILH, MILR, MILC	OK	---
	=DT, <>DT, <DT, <=DT, >DT, >=DT	OK	---
	BCMP2	OK	---
	GRY	OK	OK from lot number 030201
	TPO	OK	---
	DSW, TKY, HKY, MTR, 7SEG	OK	---
	EXPLT, EGATR, ESATR, ECHRD, ECHWR	OK	---
	Reading/Writing CPU Bus Units with IORD/IOWR	OK	OK from lot number 030418
	PRV2	---	---

CS1D CPU Units

Function		CS1D CPU Units for Single-CPU Systems (CS1D-CPU□□S)	CS1D CPU Units for Duplex-CPU Systems (CS1D-CPU□□H)	
		Unit version 2.0	Unit version 1.1 or later	Pre-Ver. 1.1
Functions unique to CS1D CPU Units	Duplex CPU Units	---	OK	OK
	Online Unit Replacement	OK	OK	OK
	Duplex Power Supply Units	OK	OK	OK
	Duplex Controller Link Units	OK	OK	OK
	Duplex Ethernet Units	---	OK	OK
	Unit removal without a Programming Device	---	OK (Unit version 1.2 or later)	---
Downloading and Uploading Individual Tasks		OK	---	---
Improved Read Protection Using Passwords		OK	---	---
Write Protection from FINS Commands Sent to CPU Units via Networks		OK	---	---
Online Network Connections without I/O Tables		OK	---	---
Communications through a Maximum of 8 Network Levels		OK	---	---
Connecting Online to PLCs via NS-series PTs		OK	---	---
Setting First Slot Words		OK for up to 64 groups	---	---
Automatic Transfers at Power ON without a Parameter File		OK	---	---
Automatic Detection of I/O Allocation Method for Automatic Transfer at Power ON		---	---	---
Operation Start/End Times		OK	OK	---
New Application Instructions	MILH, MILR, MILC	OK	---	---
	=DT, <>DT, <DT, <=DT, >DT, >=DT	OK	---	---
	BCMP2	OK	---	---
	GRY	OK	---	---
	TPO	OK	---	---
	DSW, TKY, HKY, MTR, 7SEG	OK	---	---
	EXPLT, EGATR, ESATR, ECHRD, ECHWR	OK	---	---
	Reading/Writing CPU Bus Units with IORD/IOWR	OK	---	---
	PRV2	OK	---	---

CJ1-H/CJ1M CPU Units

Function		CJ1-H CPU Units		CJ1M CPU Units		
		CJ1H-CPU□□H-R CJ1□-CPU□□H CJ1G-CPU□□P		CJ1M-CPU12/13/22/23		CJ1M-CPU11/21
		Unit version 2.0 or later	Other unit versions	Unit version 2.0 or later	Other unit versions	Other unit versions
Downloading and Uploading Individual Tasks		OK	---	OK	---	OK
Improved Read Protection Using Passwords		OK	---	OK	---	OK
Write Protection from FINS Commands Sent to CPU Units via Networks		OK	---	OK	---	OK
Online Network Connections without I/O Tables		OK	--- (Supported if I/O tables are automatically generated at startup.)	OK	--- (Supported if I/O tables are automatically generated at startup.)	OK
Communications through a Maximum of 8 Network Levels		OK	---	OK	---	OK
Connecting Online to PLCs via NS-series PTs		OK	OK from lot number 030201	OK	OK from lot number 030201	OK
Setting First Slot Words		OK for up to 64 groups	OK for up to 8 groups	OK for up to 64 groups	OK for up to 8 groups	OK for up to 64 groups
Automatic Transfers at Power ON without a Parameter File		OK	---	OK	---	OK
Automatic Detection of I/O Allocation Method for Automatic Transfer at Power ON		---	---	---	---	---
Operation Start/End Times		OK	---	OK	---	OK
New Application Instructions	MILH, MILR, MILC	OK	---	OK	---	OK
	=DT, <>DT, <DT, <=DT, >DT, >=DT	OK	---	OK	---	OK
	BCMP2	OK	---	OK	OK	OK
	GRY	OK	OK from lot number 030201	OK	OK from lot number 030201	OK
	TPO	OK	---	OK	---	OK
	DSW, TKY, HKY, MTR, 7SEG	OK	---	OK	---	OK
	EXPLT, EGATR, ESATR, ECHRD, ECHWR	OK	---	OK	---	OK
	Reading/Writing CPU Bus Units with IORD/IOWR	OK	---	OK	---	OK
	PRV2	---	---	OK, but only for CPU Units with built-in I/O	---	OK, but only for CPU Units with built-in I/O

User programs that contain functions supported only by CPU Units with unit version 2.0 or later cannot be used on CS/CJ-series Pre-Ver. 2.0 CPU Units. An error message will be displayed if an attempt is made to download programs containing unit version s.0 functions to a Pre-Ver. 2.0 CPU Unit, and the download will not be possible.

If an object program file (.OBJ) using these functions is transferred to a Pre-Ver. 2.0 CPU Unit, a program error will occur when operation is started or when the unit version 2.0 function is executed, and CPU Unit operation will stop.

Unit Versions and Programming Devices

The following tables show the relationship between unit versions and CX-Programmer versions.

Unit Versions and Programming Devices

CPU Unit	Functions (See note 1.)		CX-Programmer				Programming Console
			Ver. 3.3 or lower	Ver. 4.0	Ver. 5.0 Ver. 6.0	Ver. 7.0 or higher	
CS/CJ-series unit version 4.0	Functions added for unit version 4.0	Using new functions	---	---	---	OK (See note 2 and 3.)	No restrictions
		Not using new functions	OK	OK	OK	OK	
CS/CJ-series unit version 3.0	Functions added for unit version 3.0	Using new functions	---	---	OK	OK	
		Not using new functions	OK	OK	OK	OK	
CS/CJ-series unit version2.0	Functions added for unit version 2.0	Using new functions	---	OK	OK	OK	
		Not using new functions	OK	OK	OK	OK	
CS1D CPU Units for Single-CPU Systems, unit version 2.0	Functions added for unit version 2.0	Using new functions	---	OK	OK	OK	
		Not using new functions					
CS1D CPU Units for Duplex-CPU Systems, unit version 1.	Functions added for unit version 1.1	Using function blocks	---	OK	OK	OK	
		Not using function blocks	OK	OK	OK	OK	

Note

1. As shown above, there is no need to upgrade to CX-Programmer version as long as the functions added for unit versions are not used.
2. CX-Programmer version 7.1 or higher is required to use the new functions of the CJ1-H-R CPU Units. CX-Programmer version 7.22 or higher is required to use unit version 4.1 of the CJ1-H-R CPU Units. You can check the CX-Programmer version using the ***About*** menu command to display version information.
3. CX-Programmer version 7.0 or higher is required to use the functional improvements made for unit version 4.0 of the CS/CJ-series CPU Units. With CX-Programmer version 7.2 or higher, you can use even more expanded functionality.

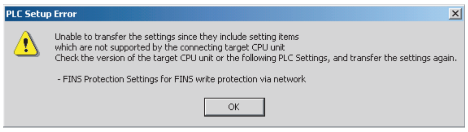
Device Type Setting

The unit version does not affect the setting made for the device type on the CX-Programmer. Select the device type as shown in the following table regardless of the unit version of the CPU Unit.

Series	CPU Unit group	CPU Unit model	Device type setting on CX-Programmer Ver. 4.0 or higher
CS Series	CS1-H CPU Units	CS1G-CPU□□H	CS1G-H
		CS1H-CPU□□H	CS1H-H
	CS1D CPU Units for Duplex-CPU Systems	CS1D-CPU□□H	CS1D-H (or CS1H-H)
	CS1D CPU Units for Single-CPU Systems	CS1D-CPU□□S	CS1D-S
CJ Series	CJ1-H CPU Units	CJ1G-CPU□□H CJ1G-CPU□□P	CJ1G-H
		CJ1H-CPU□□H-R (See note.) CJ1H-CPU□□H	CJ1H-H
	CJ1M CPU Units	CJ1M-CPU□□	CJ1M

Note Select one of the following CPU types: CPU67-R, CPU66-R, CPU65-R, or CPU64-R.

Troubleshooting Problems with Unit Versions on the CX-Programmer

Problem	Cause	Solution
 Unable to download program(s). Errors found during compilation After the above message is displayed, a compiling error will be displayed on the *Compile* Tab Page in the Output Window.	An attempt was made to download a program containing instructions supported only by later unit versions or a CPU Unit to a previous unit version.	Check the program or change to a CPU Unit with a later unit version.
 Unable to transfer the settings since they include setting items which are not supported by the connecting target CPU unit Check the version of the target CPU unit or the following PLC Settings, and transfer the settings again. - FINS Protection Settings for FINS write protection via network	An attempt was made to download a PLC Setup containing settings supported only by later unit versions or a CPU Unit to a previous unit version.	Check the settings in the PLC Setup or change to a CPU Unit with a later unit version.
"????" is displayed in a program transferred from the PLC to the CX-Programmer.	An attempt was made to upload a program containing instructions supported only by higher versions of CX-Programmer to a lower version.	New instructions cannot be uploaded to lower versions of CX-Programmer. Use a higher version of CX-Programmer.

Related Manuals

Name	Cat. No.	Contents
SYSMAC CS/CJ/NSJ Series CJ2H-CPU6□-EIP, CJ2H-CPU6□, CJ2M-CPU□□, CS1G/H-CPU□□H, CS1G/H-CPU□□-EV1, CS1D-CPU□□HA, CS1D-CPU□□SA, CS1D-CPU□□P, CS1D-CPU□□H, CS1D-CPU□□S, CJ1H-CPU□□H-R, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□, CJ1G-CPU□□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Programmable Controllers Instructions Reference Manual	W474 (this manual)	Provides detailed descriptions of the instructions. When programming, use this manual together with the manuals for your CPU Unit.
CJ Series CJ2 CPU Unit Hardware User's Manual CJ2H-CPU6□-EIP, CJ2H-CPU6□, CJ2M-CPU□□	W472	Provides the following information on the CJ2 CPU Units: Overview, system design, hardware specifications, hardware settings, installation, wiring, maintenance, and troubleshooting. Use this manual together with the *CJ2 CPU Unit Software User's Manual* (W473).
CJ Series CJ2 CPU Unit Software User's Manual CJ2H-CPU6□-EIP, CJ2H-CPU6□, CJ2M-CPU□□	W473	Provides the following information on the CJ2 CPU Units: Overview of CPU Unit operation, programming, software settings, CPU Unit functions, and system startup. Use this manual together with the *CJ2 CPU Unit Hardware User's Manual* (W472).
CJ-series CJ2M CPU Unit Pulse I/O Module User's Manual CJ2M-CPU□□ + CJ2M-MD21□	W486	Provides the following information on the Pulse I/O Module for CJ2M CPU Units: • Specifications and wiring methods • I/O functions • Quick-response inputs • Interrupt functions • High-speed counters • Pulse outputs • PWM outputs When programming, use this manual together with the *Instructions Reference Manual* (Cat. No. W474).
SYSMAC CS/CJ/NSJ Series CS1G/H-CPU□□H, CS1G/H-CPU□□-EV1, CS1D-CPU□□HA, CS1D-CPU□□SA, CS1D-CPU□□P, CS1D-CPU□□H, CS1D-CPU□□S, CJ1H-CPU□□-H-R, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□, CJ1G-CPU□□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Programmable Controllers Programming Manual	W394	Describes programming, tasks, file memory, and other functions for the CS-series, CJ-series, and NSJ-series PLCs. Use this manual together with the *CS-series Programmable Controller Operation Manual* (W339) ,*CJ-series Programmable Controller Operation Manual* (W393) or *CS1D CPU Units* (W394).
SYSMAC CS Series CS1G/H-CPU□□H Programmable Controllers Operation Manual	W339	Provides an outline of, and describes the design, installation, maintenance, and other basic operations for the CS-series PLCs. Information is also included on features, system configuration, wiring, I/O memory allocations, and troubleshooting. Use this manual together with the *Programmable Controllers Programming Manual* (W394).

Name	Cat. No.	Contents
SYSMAC CJ Series CJ1H-CPU□□H-R, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□, CJ1G-CPU□□ Programmable Controllers Operation Manual	W393	Provides an outline of, and describes the design, installation, maintenance, and other basic operations for the CJ-series PLCs. Information is also included on features, system configuration, wiring, I/O memory allocations, and troubleshooting. Use together with the *Programmable Controllers Programming Manual* (W394).
SYSMAC CS/CJ/CP/NSJ Series CJ2H-CPU6□-EIP, CJ2H-CPU6□, CJ2M-CPU□□, CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CS1D-CPU□□HA, CS1D-CPU□□SA, CS1D-CPU□□P, CS1D-CPU□□H, CS1D-CPU□□S, CJ1H-CPU□□H-R, CJ1G-CPU□□, CJ1G-CPU□□P, CJ1M-CPU□□, CJ1G/H-CPU□□H, CS1W-SCB□□-V1, CS1W-SCU□□-V1, CJ1W-SCU□□-V1, CP1H-X□□□□-□, CP1H-XA□□□□-□, CP1H-Y□□□□-□, CP1L-M/L□□□-□, CP1E-E/N□□D□-□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Communications Commands Reference Manual	W342	Describes the C-series (Host Link) and FINS communications commands used with CS/CJ-series PLCs.
CXONE-AL□□D-V4 CX-Programmer Operation Manual	W446	Provides information on how to use the CX-Programmer for all functionality except for function blocks.
SYSMAC CS Series CS1D-CPU□□HA CPU Units CS1D-CPU□□SA CPU Units CS1D-CPU□□P CPU Units CS1D-CPU□□H CPU Units CS1D-CPU□□S CPU Units CS1D-DPL01/02D Duplex Unit CS1D-PA/PD□□□ Power Supply Unit Duplex System Operation Manual	W405	Provides an outline of and describes the design, installation, maintenance, and other basic operations for a Duplex System based on CS1D CPU Units.
NSJ Series NSJ5-TQ□□(B)-G5D, NSJ5-SQ□□(B)-G5D, NSJ8-TV□□(B)-G5D, NSJ10-TV□□(B)-G5D, NSJ12-TS□□(B)-G5D, NSJ5-TQ□□(B)-M3D, NSJ5-SQ□□(B)-M3D, NSJ8-TV□□(B)-M3D, NSJW-ETN21, NSJW-CLK21-V1, NSJW-IC101 Operation Manual	W452	Provides the following information on the NSJ-series NSJ Controllers: Overview and features Designing the system configuration Installation and wiring I/O memory allocations Troubleshooting and maintenance Use this manual in combination with the following manuals: SYSMAC CS Series Operation Manual (W339), SYSMAC CJ Series Operation Manual (W393), SYSMAC CS/CJ Series Programming Manual (W394), and NS-V1/-V2 Series Setup Manual (V083)
CXONE-AL□□D-V4 CX-Programmer Operation Manual Function Blocks/Structured Text	W447	Describes the functionality unique to the CX-Programmer and CP-series CPU Units or CS/CJ-series CPU Units with unit version 3.0 or later based on function blocks or structured text programming. Functionality that is the same as that of the CX-Programmer is described in W446 (enclosed).
SYSMAC CS/CJ Series CQM1H-PRO01-E, C200H-PRO27-E, CQM1-PRO01-E, CS1W-KS001 Programming Consoles Operation Manual	W341	Provides information on how to program and operate CS/CJ-series PLCs using a Programming Console.

Name	Cat. No.	Contents
SYSMAC CS/CJ Series CS1W-SCB□□-V1, CS1W-SCU□□-V1, CJ1W-SCU□□-V1, CJ1W-SCU□2 Serial Communications Boards/Units Operation Manual	W336	Describes the use of Serial Communications Unit and Boards to perform serial communications with external devices, including the usage of standard system protocols for OMRON products.
CXONE-AL□□D-V4 CX-Protocol Operation Manual	W344	Describes the use of the CX-Protocol to create protocol macros as communications sequences to communicate with external devices.
CXONE-AL□□D-V4 CX-Integrator Operation Manual	W464	Describes operating procedures for the CX-Integrator Network Configuration Tool for CS-, CJ-, CP-, and NSJ-series Controllers.
CXONE-AL□□D-V4, CXONE-LT□□□-V4 CX-One Setup Manual	W463	Installation and overview of CX-One FA Integrated Tool Package.

⚠ **WARNING** Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each section in its entirety and be sure you understand the information provided in the section and related sections before attempting any of the procedures or operations given.

Safety Precautions

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.

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.

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.

Safety Precautions

⚠ WARNING Do not attempt to take any Unit apart or touch the inside of any Unit 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 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 Programmable Controller or another external factor affecting the operation of the Programmable Controller. "Programmable Controller" indicates the CPU Unit and all other Units and is abbreviated "PLC" in this manual. 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. Unexpected operation, however, may still occur for errors in the I/O control section, errors in I/O memory, and other errors that cannot be detected by the self-diagnosis function. As a countermeasure for all such errors, external safety measures must be provided to ensure safety in the system.
- The PLC outputs may remain ON or OFF due to deposition 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.
- Provide measures in the computer system and programming to ensure safety in the overall system even if errors or malfunctions occur in data link communications or remote I/O communications.

⚠ **Caution** Confirm safety before transferring data files stored in the file memory (Memory Card or EM file memory) to the I/O area (CIO) of the CPU Unit using a peripheral tool. Otherwise, the devices connected to the Output Unit may malfunction regardless of the operating mode of the CPU Unit.

⚠ **Caution** 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. Serious accidents may result from abnormal operation if proper measures are not provided.

⚠ **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, PLC Setup, I/O tables, I/O memory contents, or parameters to another node or changing contents of the any of these items. Transferring or changing data can result in unexpected system operation.

⚠ **Caution** The CS1-H/CJ1-H/CJ1M/CS1D/CJ2 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, EM, HR Areas, and the variables set to be held in function blocks), however, is not written to flash memory. The DM, EM, HR Areas, and the variables set to be held in function blocks 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, EM, HR Areas, and the variables set to be held in function blocks are used to control external outputs, prevent inappropriate outputs from being made whenever the Battery Error Flag (A402.04) is ON.

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

⚠ **Caution** Do not touch the Power Supply Unit when power is being supplied or immediately after the power supply is turned OFF. The Power Supply Unit will be hot and you may be burned.

⚠ **Caution** Be careful when connecting personal computers or other peripheral devices to a PLC to which is mounted a non-insulated Unit (CS1W-CLK13/53/12/52(-V1) or CS1W-ETN01) connected to an external power supply. A short-circuit will be created if the 24 V side of the external power supply is grounded and the 0 V side of the peripheral device is grounded. When connecting a peripheral device to this type of PLC, either ground the 0 V side of the external power supply or do not ground the external power supply at all.

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. Be sure that the operating environment is within the specified conditions at installation and remains within the specified conditions during the life of the system.

Application Precautions

Observe the following precautions when using a CJ-series PLC.

■ Power Supply

- Always use the power supply voltages specified in the operation manuals. An incorrect voltage may result in malfunction or burning.
- Design the system so that the power supply capacity of the Power Supply Unit that you are using is not exceeded. Exceeding the capacity of the Power Supply Unit may prevent the CPU Unit or other Units from starting.
- 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.
- 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 Power Supply Units, I/O Units, CPU Units, Option Boards, Pulse I/O Modules, or any other Units.
  - Assembling the Units.
  - Setting DIP switches or rotary switches.
  - Connecting cables or wiring the system.
  - Connecting or disconnecting the connectors.
- When cross-wiring terminals, the total current for all the terminal will flow in the wire. Make sure that the current capacity of the wire is sufficient.

- Observe the following precautions when using a Power Supply Unit that supports the Replacement Notification Function.
  - Replace the Power Supply Unit within six months if the display on the front of the Power Supply Unit alternates between 0.0 and A02, or if the alarm output automatically turns OFF.
  - Keep the alarm output cable separated from power line and high-voltage lines.
  - Do not apply a voltage or connect a load exceeding the specifications to the alarm output.
  - When storing the Power Supply Unit for more than three months, store it at −20 to 30°C and 25% to 70% humidity to preserve the Replacement Notification Function.
  - If the Power Supply Unit is not installed properly, heat buildup may cause the replacement notification signal to appear at the wrong time or may cause interior elements to deteriorate or become damaged. Use only the standard installation method.
- Do not touch the terminals on the Power Supply Unit immediately after turning OFF the power supply. Residual voltage may cause electrical shock.

■ Installation

- Do not install the PLC near sources of strong high-frequency noise.
- Before touching a Unit, be sure to first touch a grounded metallic object in order to discharge any static build-up. Not doing so may result in malfunction or damage.
- Be sure that the terminal blocks, connectors, Memory Cards, Option Boards, Pulse I/O Modules, expansion cables, and other items with locking devices are properly locked into place.
- The sliders on the tops and bottoms of the Power Supply Unit, CPU Unit, I/O Units, Special I/O Units, CPU Bus Units, and Pulse I/O Modules must be completely locked (until they click into place) after connecting to adjacent Units.

■ Wiring

- Follow the instructions in hardware manual for your PLC to correctly perform wiring.
- Double-check all wiring and switch settings before turning ON the power supply. Incorrect wiring may result in burning.
- Be sure that all terminal screws, and cable connector screws are tightened to the torque specified in the relevant manuals.
- Mount terminal blocks and connectors only after checking the mounting location carefully.
- Leave the label attached to the Unit when wiring. Removing the label may result in malfunction if foreign matter enters the Unit.
- 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 Units in excess of the rated input voltage. Excess voltages may result in burning.
- Always connect to a ground of 100 Ω or less when installing the Units. Not connecting to a ground of 100 Ω or less may result in electric shock.
  A ground of 100 Ω or less must be installed when shorting the GR and LG terminals on the Power Supply Unit.

- Do not apply voltages or connect loads to the Output Units in excess of the maximum switching capacity. Excess voltage or loads may result in burning.
- 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 or other wiring lines. Doing so may break the cables.
- Do not use commercially available RS-232C personal computer cables. Always use the special cables listed in this manual or make cables according to manual specifications. Using commercially available cables may damage the external devices or CPU Unit.
- Never connect pin 6 (5-V power supply) on the RS-232C port on the CPU Unit to any device other than an NT-AL001 or CJ1W-CIF11 Adapter. The external device or the CPU Unit may be damaged.

■ Handling

- The Power Supply Unit may possibly be damaged if the entire voltage for a dielectric strength test is applied or shut OFF suddenly using a switch. Use a variable resistor to gradually increase and decrease the voltage.
- Separate the line ground terminal (LG) from the functional ground terminal (GR) on the Power Supply Unit before performing withstand voltage tests or insulation resistance tests. Not doing so may result in burning.
- Make sure that the DIP switches and DM Area are set correctly before starting operation.
- After replacing the CPU Unit, a Special I/O Unit, or a CPU Bus Unit, make sure that the required data for the DM Area, Holding Area, and other memory areas has been transferred to the new Unit before restarting 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 attempt to disassemble, repair, or modify any Units. Any attempt to do so may result in malfunction, fire, or electric shock.
- Do not drop the PLC or subject abnormal vibration or shock to it.
- The life of the battery will be reduced if the PLC is left for a period of time without a battery installed and without power supply, and then a battery is installed without turning ON the power supply.
- Replace the battery as soon as a battery error occurs or as soon as the specified battery backup time expires. Be sure to install a replacement battery within two years of the production date shown on the battery's label.
- Before replacing the battery, turn ON power for at least 5 minutes before starting the replacement procedure and complete replacing the battery within 5 minutes of turning OFF the power supply. Memory contents may be corrupted if this precaution is not obeyed.
- If the Battery Error Flag is used in programming the application, confirm system safety even if the system detects a battery error before you replace the battery while the power is ON.
- Do not short the battery terminals or charge, disassemble, heat, or incinerate the battery. Do not subject the battery to strong shocks. 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 oth-

erwise subjected to excessive shock. Batteries that have been subjected to shock may leak if they are used.

- UL standards require that only an experienced engineer can replace the battery. Make sure that an experienced engineer is in charge of battery replacement. Follow the procedure for battery replacement given in hardware manual for your PLC.
- Dispose of the product and batteries according to local ordinances as they apply.

廢電池請回收

- 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.)
- Unexpected operation may result if inappropriate data link tables or parameters are set. Even if appropriate data link tables and parameters have been set, confirm that the controlled system will not be adversely affected before starting or stopping data links.
- Write programs so that any data that is received for data link communications is used only if there are no errors in the CPU Units that are the sources of the data. Use the CPU Unit error information in the status flags to check for errors in the source CPU Units. If there are errors in source CPU Units, they may send incorrect data.
- All CPU Bus Units will be restarted when routing tables are transferred from a Programming Device to the CPU Unit. Restarting these Units is required to read and enable the new routing tables. Confirm that the system will not be adversely affected before transferring the routing tables.
- Tag data links between related nodes will stop while tag data link parameters are being transferred during PLC operation. Confirm that the system will not be adversely affected before transferring the tag data link parameters.
- If there is interference with network communications, output status will depend on the products that are being used. When using products with outputs, confirm the operation that will occur when there is interference with communications and implement safety measures.
- When creating an AUTOEXEC.IOM file from a Programming Device (a Programming Console or the CX-Programmer) to automatically transfer data at startup, set the first write address to D20000 and be sure that the size of data written does not exceed the size of the DM Area. When the data file is read from the Memory Card at startup, data will be written in the CPU Unit starting at D20000 even if another address was set when the AUTOEXEC.IOM file was created. Also, if the DM Area is exceeded (which is possible when the CX-Programmer is used), the remaining data will be written to the EM Area.
- The user program and parameter area data in the CJ2 CPU Units are 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.

- Check the user program and Unit parameter settings for proper execution before actually running them on the Unit. Not checking the program and parameter settings may result in an unexpected operation.
- When setting a Special I/O Unit or CPU Bus Unit in the I/O tables, carefully check the safety of the devices at the connection target before restarting the Unit.
- If a symbol or memory address (only symbols are allowed for ST programming) is specified for the suffix of an array variable in ladder or ST programming to indirectly specify the element number, be sure that the element number does not exceed the maximum memory area range. Specifying a element number that exceeds the maximum range of the memory area specified for the symbol will result accessing data in a different memory area, and may result in unexpected operation.
- Program so that the memory area of the start address is not exceeded when using a symbol or address is used to specify the offset directly in a ladder program.
  If an indirect specification causes the address to exceed the memory area of the start address, the system will access data in other area, and unexpected operation may occur.
- A CS1 or CJ1 CPU Unit program file (.OBJ) cannot be transferred directly to a CJ2 CPU Unit using a Memory Card. It must first be converted for use with a CJ2 CPU Unit using the CX-Programmer.

■ External Circuits

- Always turn ON power to the PLC before turning ON power to the control system. 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.
- Install external breakers and take other safety measures against short-circuiting in external wiring. Insufficient safety measures against short-circuiting may result in burning.
- Do not turn OFF the power supply to the PLC when reading or writing a Memory Card. Also, do not remove the Memory Card when the BUSY indicator is lit. Doing so may make the Memory Card unusable.
  To remove a Memory Card, first press the memory card power supply switch and then wait for the BUSY indicator to go out before removing the Memory Card.

TABLE OF CONTENTS

SECTION 1
Basic Understanding of Instructions 1

SECTION 2
Instructions (Applicable CPU Units) 23

SECTION 3
Instructions 45

TABLE OF CONTENTS

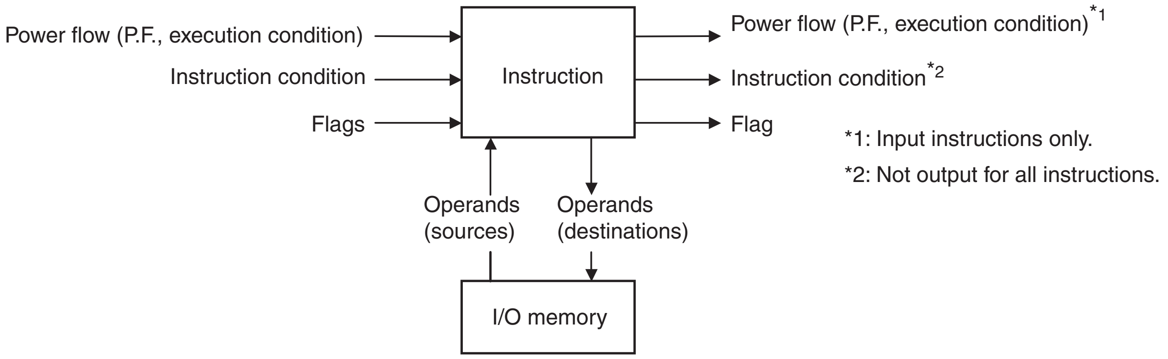
SECTION 1
Basic Understanding of Instructions

This section describes the basic information that is required to use programming instructions.

1-1 Basic Understanding of Instructions

Structure of Instructions

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



*1: Input instructions only.

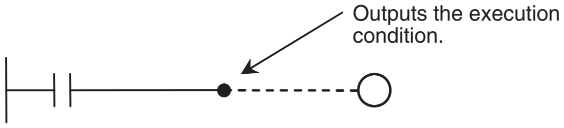
*2: Not output for all instructions.

Power Flow

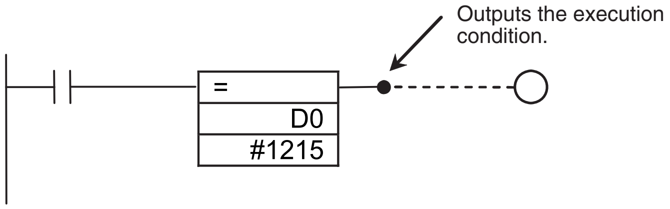
The power flow is the execution condition that is used to control the execute and instructions when programs are executing normally. In a ladder program, power flow represents the status of the execution condition.

Input Instructions

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

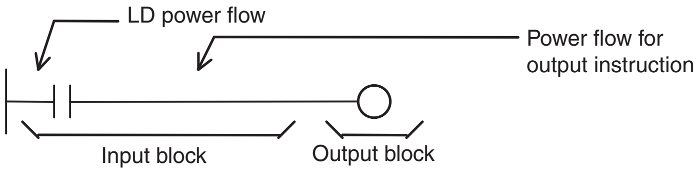


- Intermediate instructions input the power flow as an execution condition and output the power flow to an intermediate or output instruction.



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 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

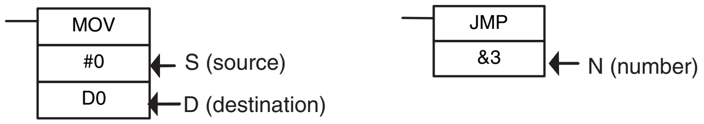
In this context, a flag is a bit that serves as an interface between instructions.

Input flags		Output flags	
Flag	**Description**	**Flag**	**Description**
Carry (CY) Flag	The Carry Flag is used as an unspecified operand in data shift instructions and addition/subtraction instructions.	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 P_On, P_Off, P_ER, P_CY, P_EQ rather than by addresses.
Flags for Special Instructions	These include teaching flags for FPD(269) instructions and network communications enabled flags.	Flags for Special Instructions	These include memory card instruction flags and MSG(046) execution completed flags.

Operands

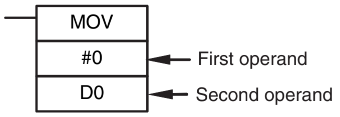
Operands specify preset instruction parameters (boxes in ladder diagrams) that are used to specify I/O memory area contents or constants. An instruction can be executed entering an address or constant as the operands. Operands are classified as source, destination, or number operands.

Example



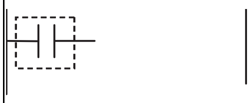
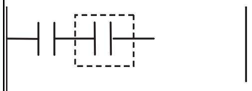
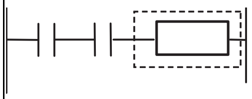
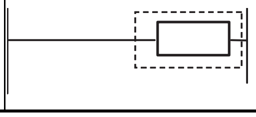
Operand types		Operand symbol	Description	
Source	Specifies the address of the data to be read or a constant.	S	Source Operand	Source operand other than control data (C)
		C	Control data	Compound data in a source operand that has different meanings depending bit status.
Destination (Results)	Specifies the address where data will be written.	D	---	
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.



Instruction Location and Execution Conditions

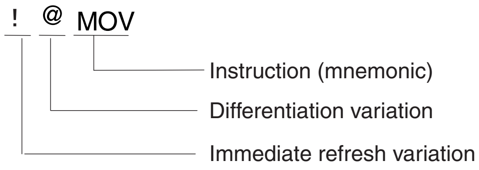
The following table shows the possible locations for instructions. Instructions are grouped into those that do and those do not require execution conditions. Refer to *SECTION 3 Instructions* for details.

Instruction		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.

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.



Execution Conditions

The following two types of basic and special instructions can be used.

- Non-differentiated instructions: Executed every cycle
- Differentiated instructions: Executed only once

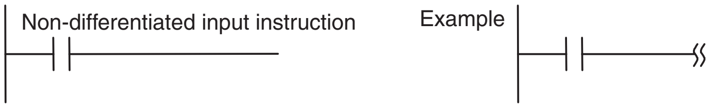
Non-differentiated Instructions

■ Output instructions (Instructions That Require Input Conditions):

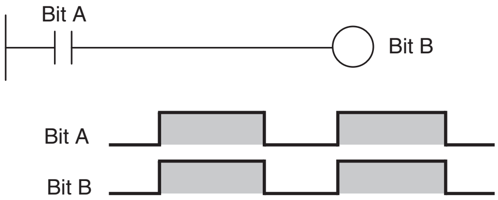
These instructions are executed once every cycle while the execution conditions are satisfied (ON or OFF).

■ Input Instructions (Logical Starts and Intermediate Instructions):

These 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).



■ Time Chart

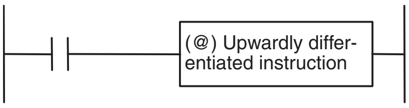


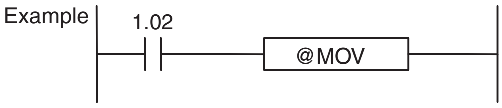
Input-differentiated Instructions

■ Upwardly Differentiated Instructions (Instruction Preceded by @)

Output Instructions:

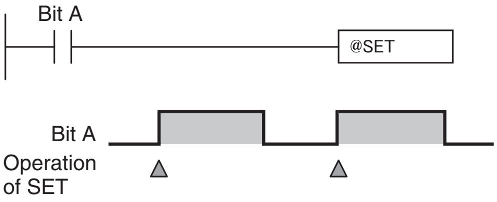
The instruction is executed only during the cycle in which the execution condition turned ON (OFF → ON) and are not executed in the following cycles. The instruction is executed only during the cycle in which the execution condition turned ON (OFF → ON) and are not executed in the following cycles.





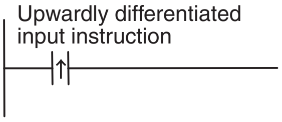
Executes the MOV instruction once when CIO 1.02 goes OFF → ON.

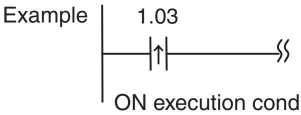
■ Time Chart



Input Instructions (Logical Starts and Intermediate Instructions):

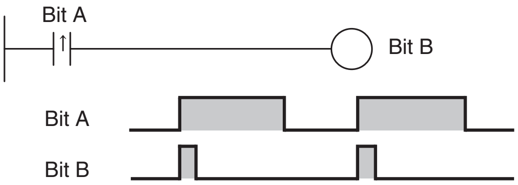
The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output an ON execution condition (power flow) when results switch from OFF to ON. The execution condition will turn OFF the next cycle. The instruction reads bit status, makes comparisons,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





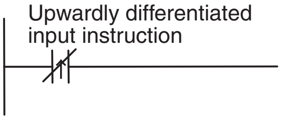
ON execution condition created for one cycle only when CIO 1.03 goes from OFF to ON.

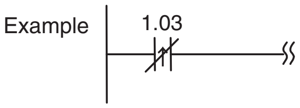
■ Time Chart



Input Instructions (Logical Starts and Intermediate Instructions):

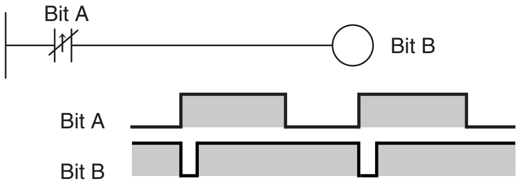
The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output an OFF execution condition (power flow stops) when results switch from OFF to ON. The execution condition will turn ON the next cycle.





OFF execution condition created for one cycle only when CIO 1.03 goes from OFF to ON.

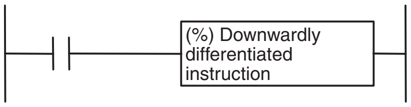
■ Time Chart



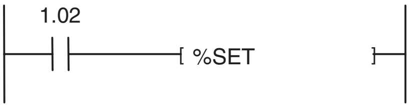
■ Downwardly Differentiated Instructions (Instruction Preceded by %)

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.

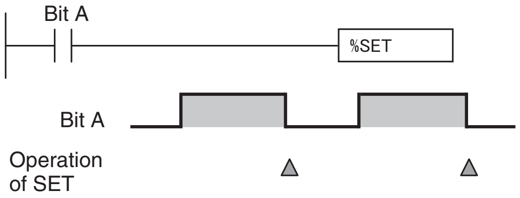


Example



Executes the SET instruction once when CIO 1.02 goes ON to OFF.

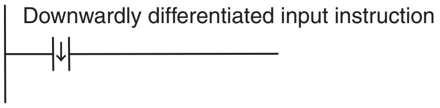
■ Time Chart

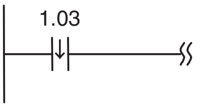


Input Instructions (Logical Starts and Intermediate Instructions):

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.

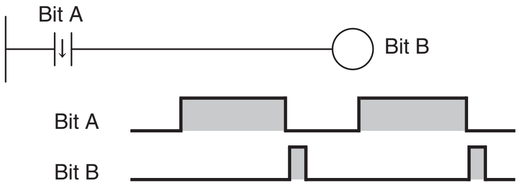
Example





Will turn ON when the CIO 1.03 switches from ON → OFF and will turn OFF after one cycle.

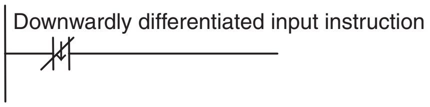
■ Time Chart



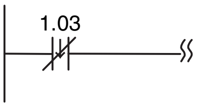
Note The downward differentiation variation (indicated by "%") cannot be used with as many instructions as the upward differentiation variation. It can be used only with the LD, AND, OR, SET, and RSET instruction. To use downward differentiation with any other instructions, combine the instruction with the DIFD(014) or DOWN(522) instruction. NOT can be added to instruction only for the CS1-H, CJ1-H, CJ1M, and CS1D CPU Units

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.

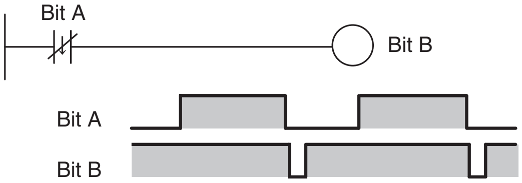


Example



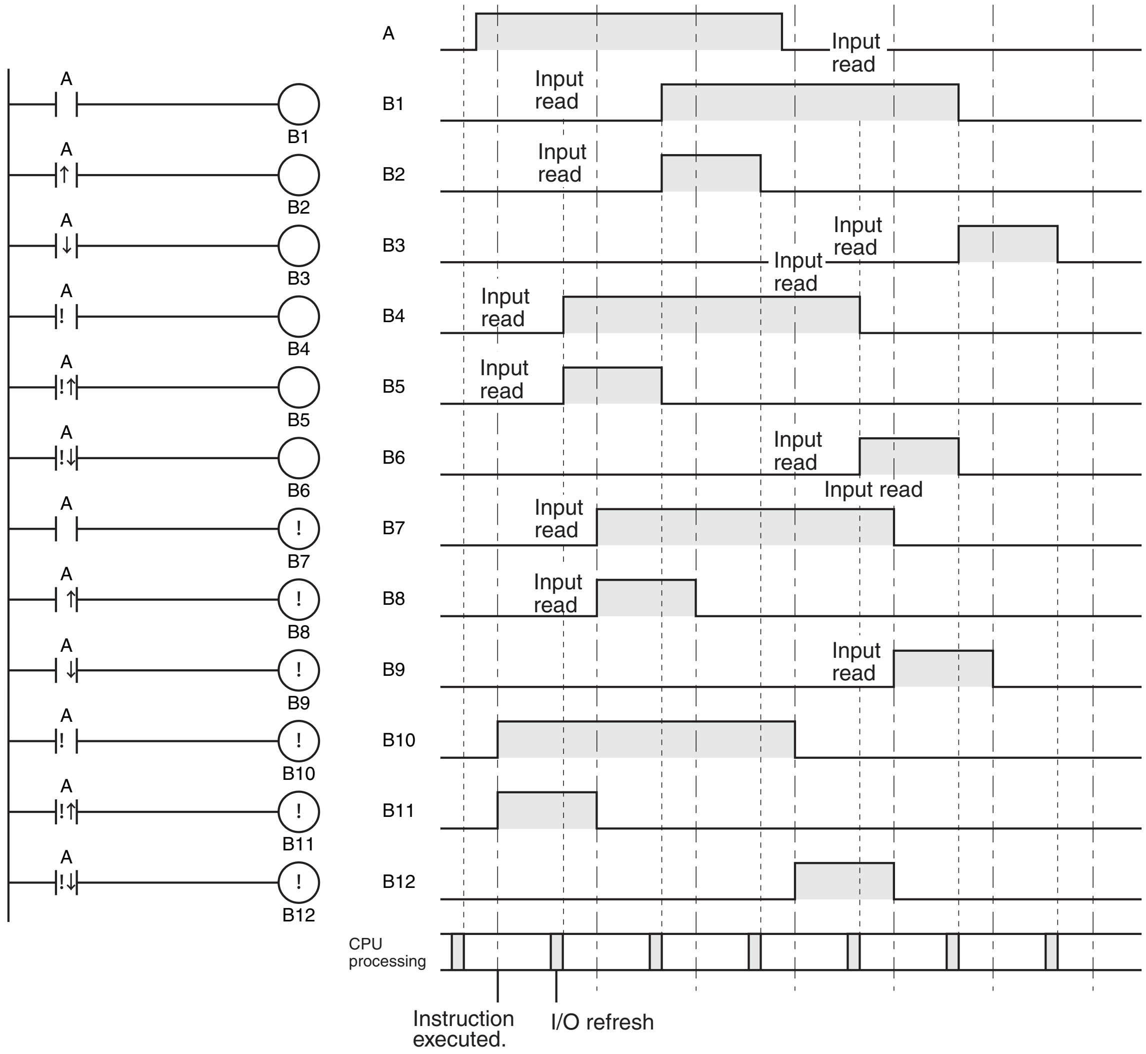
OFF execution condition created for one cycle only when CIO 1.03 goes from ON to OFF.

■ Time Chart



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.

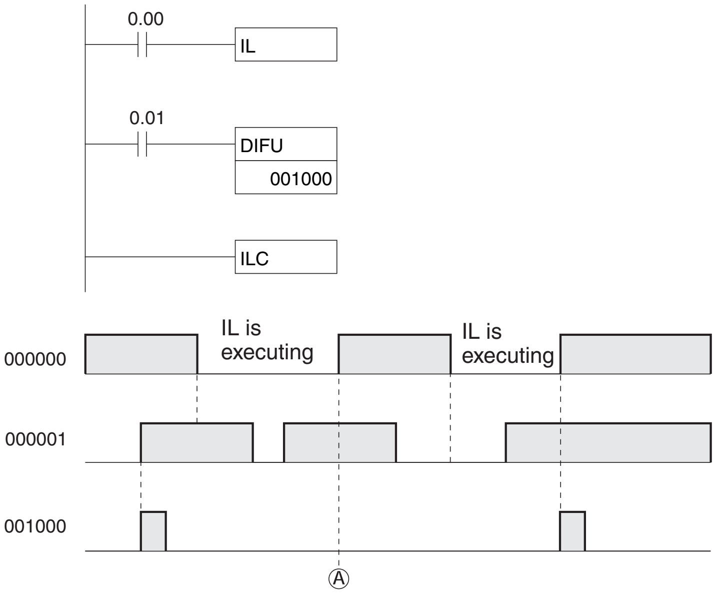


Differentiated Instructions

- 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.

■ Using Differentiated Instructions 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.



■ Using Differentiated Instructions in Jumps (JMP(004) - JME(005) 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.

- With downwardly differentiated instructions (DIFD(014) or instructions with a %), outputs will turn ON when inputs turn OFF only when the previous value flag is ON.
- With both upwardly and downwardly differentiated instructions, outputs will turn OFF in the next cycle.

Note Do not use the Always P_On Flag or A200.11 (First Cycle Flag) as the input bit for an upwardly differentiated instruction. Do not use the Always P_Off Flag as the input bit for a downwardly differentiated instruction. If either is used, the instruction will never be executed.

1-2 Specifying Operands

Addressing I/O Memory Areas

Bit Addresses

□□□□.□□

Bit number (00 to 15)

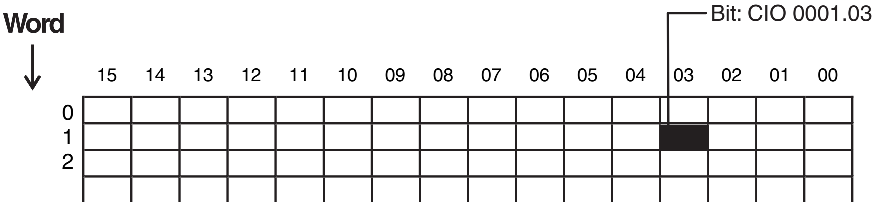
Word address

Example: The address of bit 03 in word 1 in the CIO Area would be as shown below.

1. 03

Bit number: 03

Word address: 0001



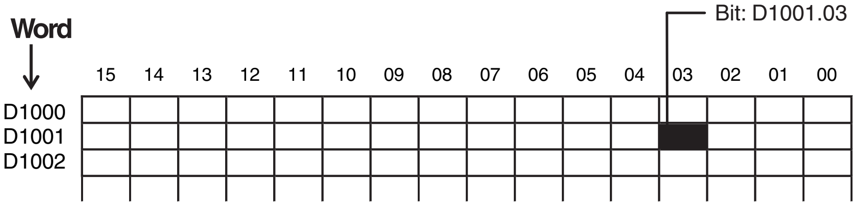
With the CJ2 CPU Unit, bit addresses can be specified in the DM and EM Areas.

Example: DM area

D1001.03

Bit number: 03

Word address: D1001



Word Addresses

□□□□

Indicates the word address

Example: I/O Area
The address for word CIO 10 would be as shown below.

10

Word address

DM and EM Areas addresses are given "D" or "E" prefixes, as shown below for the address D200.

Example: DM Area
The address for word D200 would be as shown below.

D200

Word address

Example: EM Area
The address for word E200 in the current bank would be as shown below.

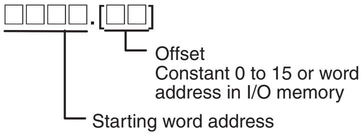
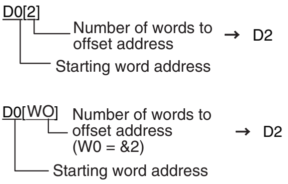
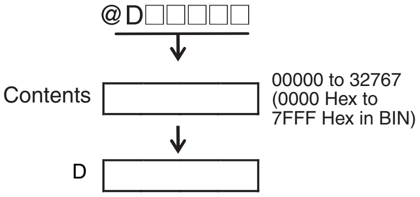
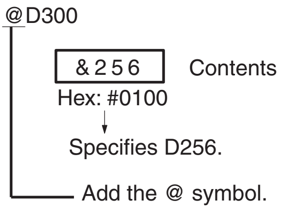
E200
Word address

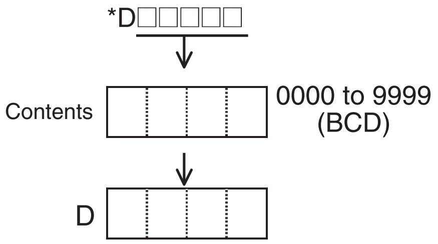
Example: EM Area Bank 1
The address for word E200 in bank 1 would be as shown below.

E01_200
Word address
Bank number

Specifying Operands

Operand	Description	Notation	Application examples
Specifying bit addresses	The word and bit numbers are specified directly to specify a bit (input bits). □□□□.□□ Bit number (00 to 15) Indicates the word address. The same addresses are used to access timer/counter Completion Flags and Present Values. There is also only one address for a Task Flag.	1.02 Bit number (02) Word number: 1	1.02 —| |—
Specifying word addresses	The word number is specified directly to specify the 16-bit word. □□□□ Indicates the word address.	3 Word number: 3 D200 Word number: 200	MOV 3 D200
Specifying offsets for bit addresses[*1]	In brackets, specify the number of bits to offset the specified starting bit address. □□□□.□□ [□□] Offset Constant 0 to 15 or word address in I/O memory Starting bit address A symbol can also be specified for the starting bit address. Only CIO, Holding, Work, DM, and EM Area addresses can be used regardless of whether a physical address or symbol is used. A constant or word address in I/O memory can be used for the offset. If a word address is specified, the contents of the word is used directly as the offset.	10.00[2] Number of bits to offset the address → 10.02 Starting bit address 10.00[WO] Number of bits to offset the address (W0 = &2) → 10.02 Starting bit address	10.00[2] —| |—

Operand	Description	Notation	Application examples
Specifying offsets for word addresses[*1]	In brackets, specify the number of bits to offset the specified starting bit address.  A symbol can also be specified for the starting word address. Only CIO, Holding, Work, DM, and EM Area addresses can be used regardless of whether a physical address or symbol is used. A constant or word address in I/O memory can be used for the offset. If a word address is specified, the contents of the word is used directly as the offset.		MOV 3 D0[200]
Specifying indirect DM/EM 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) or Extended Data Memory (EM). Add the @ symbol at the front to specify an indirect ad-dress in Binary Mode.  D0 to D32767 are specified if □D(□□□□□) contains 0000 Hex to 7FFF Hex (0 to 32767).		MOV #0001 @D300

Operand	Description	Notation	Application examples
Specifying indirect DM/EM addresses in Binary Mode	E0 _0 to E0 _32767 of bank 0 in Extended Data Memory (EM) are specified if @D(□□□□□) contains 8000 Hex to FFFF Hex (32768 to 65535).	@D300 &32769 Contents Hex: #8001 ↓ Specifies E0_01.	
	E□_0 to E□_32767 in the specified bank are specified if @E□_□□□□□ contains 0000 Hex to 7FFF Hex (0 to 32767).	@E1_200 &257 Contents Hex: #0101 ↓ Specifies E1_257.	MOV #0001@E1_200
	E(□+1)_0 to E(□+1)_32767 in the bank following the specified bank □ are specified if @E□_□□□□□ contains 8000 Hex to FFFF Hex (32768 to 65535).	@E1_200 &32770 Contents Hex: #8002 ↓ Specifies E2_2.	
	Note When specifying an indirect address in Binary Mode, treat Data Memory (DM) and Extended Data Memory (EM) (banks 0 to 18) as one series of addresses. If the contents of an address with the @ symbol exceeds 32767, the address will be assumed to be an address in the Extended Data Memory (EM) continuing on from 0 in bank No. 0. Example: If the Data Memory (DM) word contains 32768, E0_0 in bank 0 in Extended Data Memory (EM) would be specified. If the Extended Data Memory (EM) bank number is specified as "n" and the contents of the word exceeds 32767, the address will be assumed to be an address in the Extended Data Memory (EM) continuing on from 0 in bank N+1. Example: If bank 2 in Extended Data Memory (EM) contains 32768, E3_0 in bank number 3 in Extended Data Memory (EM) would be specified.		
Specifying indirect DM/EM 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) or Extended Data Memory (EM). Add an asterisk (*) at the front to specify an indirect address in BCD Mode.  	*D200 #0100 Contents ↓ Specifies D100 Add an asterisk (*).	MOV #0001 *D200

Operand	Description		Notation	Application examples
Specifying a register directly (See note.)	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 1.02 IR0 Stores the PLC memory address for CIO 0010 in IR0.
			IR1	MOVR 10 IR1 Stores the PLC memory address for CIO 0010 in IR1.
Specifying an indirect address using a register (See note.)	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	LD ,IR0 Loads the bit with the PLC memory address in IR0.
			,IR1	MOV #0001 ,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	LD +5 ,IR0 Loads the bit with the PLC memory address in IR0 + 5.
			31 ,IR1	MOV #0001 +31 ,IR1 Stores #0001 in the word with the PLC memory address in IR1 + 31
	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	LD DR0 ,IR0 Loads the bit with the PLC memory address in IR0 + the value in DR0.
			DR0 ,IR1	MOV #0001 DR0 ,IR1 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++	LD ,IR0 ++ Increments the contents of IR0 by 2 after the bit with the PLC memory address in IR0 is loaded.
			,IR1+	MOV #0001 ,IR1 + 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	LD ,––IR0 After decrementing the contents of IR0 by 2, the bit with the PLC memory address in IR0 is loaded.
			,IR1	MOV #0001 ,–IR1 After decrementing the contents of IR1 by 1, #0001 is stored in the word with the PLC memory address in IR1.

Note For specific application methods, refer to the *SYSMAC CS/CJ/NSJ-series Programmable Controllers Programming Manual* (Cat. No. W394) and the *SYSMAC CJ-series CJ2 CPU Unit Software User's Manual* (Cat. No. W473).

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 #0100 D0 Stores #0100 hex (&256 decimal) in D0. +#0009 #0001 D1 Stores #000A hex (&10 decimal) in D1.
		Signed decimal	±	–32768 to +32767	MOV .100 D0 Stores .100 decimal (#FF9C hex) in D0. +–9 –1 D1 Stores –10 decimal (#FFF6 hex) in D1.
		Unsigned decimal	&	&0 to &65535	MOV &256 D0 Stores –256 decimal (#0100 hex) in D0. +&9 &1 D1 Stores –10 decimal (#000A hex) in D1.
	All BCD data or a limited range of BCD data	BCD	#	#0000 to #9999	MOV #0100 D0 Stores #0100 (BCD) in D0. +B #0009 #0001 D1 Stores #0010 (BCD) in D1.
32-bit constant	All binary data or a limited range of binary data	Unsigned binary	#	#00000000 to #FFFFFFFF	MOVL #12345678 D0 Stores #12345678 hex in D0 and D1. D1 D0 1234 5678
		Signed binary	+	–2147483648 to +2147483647	MOVL –12345678 D0 Stores –12345678 decimal in D0 and D1.
		Unsigned decimal	&	&0 to &4294967295	MOVL &12345678 D0 Stores &12345678 decimal in D0 and D1.
	All BCD data or a limited range of BCD data	BCD	#	#00000000 to #99999999	MOVL #12345678 D0 Stores #12345678 (BCD) in D0 and D1.

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$ 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.			

		Upper four digits															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Lower four digits	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	゜		

Precautions for Correct Use

The following instructions are executed even when the input conditions are OFF. Therefore, when indirect memory addresses are specified using auto-incrementing or auto-decrementing (,IR+ or ,IR-) in an operand of any of these instructions, the value in the Index Register (IR) is refreshed each cycle regardless of the input condition (increases or decreases one every cycle). This must be considered when writing a program.

Classification	Instructions
Sequence input instructions	LD, LD NOT, AND, AND NOT, OR, OR NOT, LD TST(350), LD TSTN(351), AND TST(350), AND TSTN(351), OR TST(350), OR TSTN(351)
Sequence output instructions	OUT, OUT NOT, DIFU(013), DIFD(014)
Sequence control instructions	JMP(004), FOR(512)
Timer and counter instructions	TIM/TIMX(550), TIMH(015)/TIMHX(551), TMHH(540)/TMHHX(552), TIMU(541)/TIMUX(556), TMUH(544)/TMUHX(557), TTIM(087)/TTIMX(555), TIML(542)/TIMLX(553), MTIM(533)/MTIMX(554), CNT/CNTX(546), CNTR(012)/CNTRX(548)
Comparison instructions	Symbol comparison instructions (LD, AND, OR =, etc.(function codes: 300, 305, 310, 320, and 325))
Single-precision floating-point math instructions	Single-precision floating-point data comparison (LD, AND, OR = F, etc.(function codes: 329 to 334))
Double-precision floating-point math instructions	Double-precision floating-point data comparison (LD, AND, OR = D, etc.(function codes: 335 to 340))
Block programming instructions	BPPS(811), BPRS(812), EXIT(806), EXIT(806) NOT, IF(802), IF(802) NOT, WAIT(805), WAIT(805) NOT, TIMW(813)/TIMWX(816), CNTW(814)/CNTWX(818), TMHW(815)/TMHWX(817), LEND(810), LEND(810) NOT
Text string processing instructions	STRING COMPARISON (LD, AND, OR = $, etc. (function codes: 670 to 675))

The following ladder programming examples show how the index registers are treated.

Example 1

Ladder Program:

```
LD P_Off
OUT, IR0+
```

Operation: When the PLC memory address 0.13 is stored in IR0.

The input condition is OFF (P_Off is the Always OFF Flag), so the OUT instruction sets 0.13, which is indirectly addressed by IR0, to OFF. The OUT instruction is executed, so IR0 is incremented. As a result, the PLC memory address 0.14, which was incremented by +1 in the IR0, is stored. Therefore, in the following cycle the OUT instruction turns OFF 0.14.

Example 2

Ladder Program:

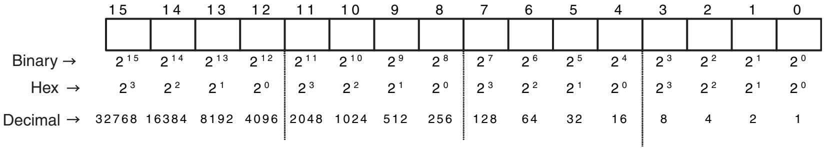
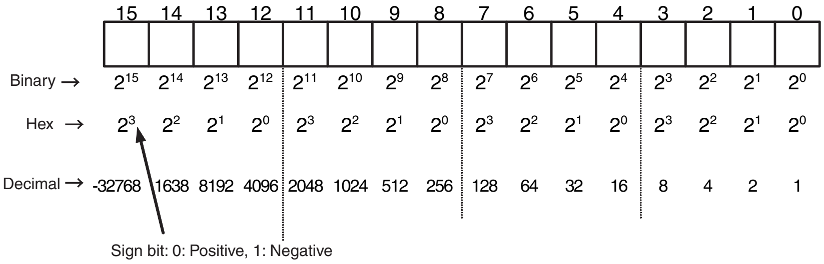
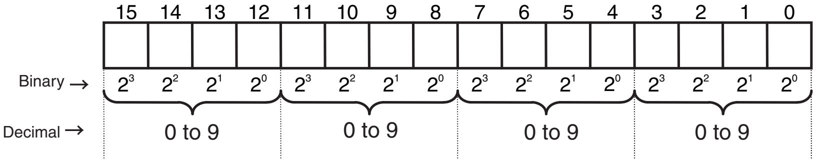
```
LD P_Off
SET, IR0+
```

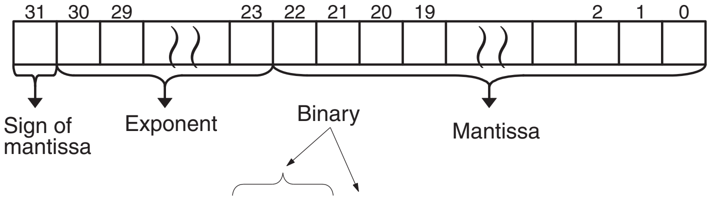
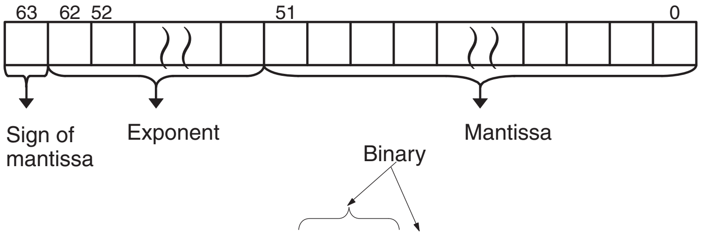
Operation: When the PLC memory address 0.13 is stored in IR0.

The input condition is OFF (P_Off is the Always OFF Flag), so the SET instruction is not executed. Therefore, IR0 is not incremented and the value stored in IR0 remains PLC memory address 0.13.

1-3 Data Formats

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

Data type	Data format	Decimal	4-digit hexadecimal
Unsigned binary		&0 to &65535	#0000 to #FFFF
Signed binary	 The data is treated as signed binary 16-bit data using the leftmost bit as the sign bit. The value is expressed in 4-digit hexadecimal. Positive numbers: If the leftmost bit is OFF, it indicates a non-negative value. For 4-digit hexadecimal, the value will be 0000 to 7FFF hex. Negative numbers: If the leftmost bit is ON, it indicates a negative value. For 4-digit hexadecimal, the value be 8000 to FFFF hex and it will be expressed as the 2's complement of the absolute value of the negative value (decimal).	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 = $(-1)^{Sign} \times 1.[Mantissa] \times 2^{Exponent}$ 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. 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 = $(-1)^{Sign} \times 1.[Mantissa] \times 2^{Exponent}$ 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. 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.	---	---

Note

1. 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.
2. 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 CS/CJ 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.

Values Represented in 1-word Data

Value (Decimal)	Binary representation			BCD representation (decimal)
	Decimal representations		Hexadecimal representation	
	Unsigned	Signed		
1	&1	+1	#0001	#0001
2	&2	+2	#0002	#0002
3	&3	+3	#0003	#0003
4	&4	+4	#0004	#0004
5	&5	+5	#0005	#0005
6	&6	+6	#0006	#0006
7	&7	+7	#0007	#0007
8	&8	+8	#0008	#0008
9	&9	+9	#0009	#0009
10	&10	+10	#000A	#0010
11	&11	+11	#000B	#0011
12	&12	+12	#000C	#0012
13	&13	+13	#000D	#0013
14	&14	+14	#000E	#0014
15	&15	+15	#000F	#0015
16	&16	+16	#0010	#0016
:	:	:	:	:
9999	&9999	+9999	#270F	#9999
10000	&10000	+10000	#2710	Not applicable.
:	:	:	:	
32767	&32767	+32767	#7FFF	
32768	&32768	Not applicable.	#8000	
:	:		:	
65535	&65535		#FFFF	
–1	Not applicable.	–1	#FFFF	Not applicable.
:		:	:	
–32768		–32768	#8000	
–32769		Not applicable.	Not applicable.	

Instructions (Applicable CPU Units)

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Sequence Input Instructions	LOAD	LD	---	Indicates a logical start and creates an ON/OFF execution condition based on the ON/OFF status of the specified operand bit.	OK	OK	OK	OK	OK	OK	OK	OK	56
		@LD	---		OK	OK	OK	OK	OK	OK	OK	OK	
		%LD	---		OK	OK	OK	OK	OK	OK	OK	OK	
		!LD	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@LD	---		OK	OK	OK	OK	OK	OK	---	OK	
		!%LD	---		OK	OK	OK	OK	OK	OK	---	OK	
	LOAD NOT	LD NOT	---	Indicates a logical start and creates an ON/OFF execution condition based on the reverse of the ON/OFF status of the specified operand bit.	OK	OK	OK	OK	OK	OK	OK	OK	56
		@LD NOT	---		OK	OK	OK	OK	OK	OK	OK	---	
		%LD NOT	---		OK	OK	OK	OK	OK	OK	OK	---	
		!LD NOT	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@LD NOT	---		OK	OK	OK	OK	OK	OK	---	---	
		!%LD NOT	---		OK	OK	OK	OK	OK	OK	---	---	
	AND	AND	---	Takes a logical AND of the status of the specified operand bit and the current execution condition.	OK	OK	OK	OK	OK	OK	OK	OK	58
		@AND	---		OK	OK	OK	OK	OK	OK	OK	OK	
		%AND	---		OK	OK	OK	OK	OK	OK	OK	OK	
		!AND	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@AND	---		OK	OK	OK	OK	OK	OK	---	OK	
		!%AND	---		OK	OK	OK	OK	OK	OK	---	OK	
	AND NOT	AND NOT	---	Reverses the status of the specified operand bit and takes a logical AND with the current execution condition.	OK	OK	OK	OK	OK	OK	OK	OK	58
		@AND NOT	---		OK	OK	OK	OK	OK	OK	OK	---	
		%AND NOT	---		OK	OK	OK	OK	OK	OK	OK	---	
		!AND NOT	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@AND NOT	---		OK	OK	OK	OK	OK	OK	---	---	
		!%AND NOT	---		OK	OK	OK	OK	OK	OK	---	---	
	OR	OR	---	Takes a logical OR of the ON/OFF status of the specified operand bit and the current execution condition.	OK	OK	OK	OK	OK	OK	OK	OK	60
		@OR	---		OK	OK	OK	OK	OK	OK	OK	OK	
		%OR	---		OK	OK	OK	OK	OK	OK	OK	OK	
		!OR	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@OR	---		OK	OK	OK	OK	OK	OK	---	OK	
		!%OR	---		OK	OK	OK	OK	OK	OK	---	OK	
	OR NOT	OR NOT	---	Reverses the status of the specified bit and takes a logical OR with the current execution condition.	OK	OK	OK	OK	OK	OK	OK	OK	60
		@OR NOT	---		OK	OK	OK	OK	OK	OK	OK	---	
		%OR NOT	---		OK	OK	OK	OK	OK	OK	OK	---	
		!OR NOT	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@OR NOT	---		OK	OK	OK	OK	OK	OK	---	---	
		!%OR NOT	---		OK	OK	OK	OK	OK	OK	---	---	
	AND LOAD	AND LD	---	Takes a logical AND between logic blocks.	OK	OK	OK	OK	OK	OK	OK	OK	62
	OR LOAD	OR LD	---	Takes a logical OR between logic blocks.	OK	OK	OK	OK	OK	OK	OK	OK	62
	NOT	NOT	520	Reverses the execution condition.	OK	OK	OK	OK	OK	OK	OK	OK	65
	CONDITION ON	UP	521	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.	OK	OK	OK	OK	OK	OK	OK	OK	66
	CONDITION OFF	DOWN	522	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.	OK	OK	OK	OK	OK	OK	OK	OK	66

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Sequence Input Instructions	LOAD BIT TEST	LD TST	350	LD TST(350) is used in the program like LD; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	68
	LOAD BIT TEST NOT	LD TSTN	351	LD TSTN(351) is used in the program like LD NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	68
	AND BIT TEST	AND TST	350	AND TST(350) is used in the program like AND; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	70
	AND BIT TEST NOT	AND TSTN	351	AND TSTN(351) is used in the program like AND NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	70
	OR BIT TEST	OR TST	350	OR TST(350) is used in the program like OR; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	72
	OR BIT TEST NOT	OR TSTN	351	OR TSTN(351) is used in the program like OR NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	72
Sequence Output Instructions	OUTPUT	OUT	---	Outputs the result (execution condition) of the logical processing to the specified bit.	OK	OK	OK	OK	OK	OK	OK	OK	74
		!OUT	---		OK	OK	OK	OK	OK	OK	---	OK	
	OUTPUT NOT	OUT NOT	---	Reverses the result (execution condition) of the logical processing, and outputs it to the specified bit.	OK	OK	OK	OK	OK	OK	OK	OK	74
		!OUT NOT	---		OK	OK	OK	OK	OK	OK	---	OK	
	TR Bits	TR	---	TR bits are used to temporarily retain the ON/OFF status of execution conditions in a program when programming in mnemonic code.	OK	OK	OK	OK	OK	OK	OK	OK	76
	KEEP	KEEP	011	Operates like a latching relay.	OK	OK	OK	OK	OK	OK	OK	OK	78
		!KEEP			OK	OK	OK	OK	OK	OK	---	OK	
	DIFFERENTIATE UP	DIFU	013	DIFU(013) turns the designated bit ON for one cycle when the execution condition goes from OFF to ON (rising edge).	OK	OK	OK	OK	OK	OK	OK	OK	82
		!DIFU			OK	OK	OK	OK	OK	OK	---	OK	
	DIFFERENTIATE DOWN	DIFD	014	DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge).	OK	OK	OK	OK	OK	OK	OK	OK	84
		!DIFD			OK	OK	OK	OK	OK	OK	---	OK	
	SET	SET	---	SET turns the operand bit ON when the execution condition is ON. After this, the specified contact will remain ON regardless of ON/OFF of the input condition.	OK	OK	OK	OK	OK	OK	OK	OK	86
		@SET	---		OK	OK	OK	OK	OK	OK	OK	OK	
		%SET	---		OK	OK	OK	OK	OK	OK	OK	OK	
		!SET	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@SET	---		OK	OK	OK	OK	OK	OK	---	OK	
		!%SET	---		OK	OK	OK	OK	OK	OK	---	OK	
	RESET	RSET	---	RSET turns the operand bit OFF when the execution condition is ON. After this, the specified contact will remain OFF regardless of ON/OFF of the input condition.	OK	OK	OK	OK	OK	OK	OK	OK	86
		@RSET	---		OK	OK	OK	OK	OK	OK	OK	OK	
		%RSET	---		OK	OK	OK	OK	OK	OK	OK	OK	
		!RSET	---		OK	OK	OK	OK	OK	OK	---	OK	
		!@RSET	---		OK	OK	OK	OK	OK	OK	---	OK	
		!%RSET	---		OK	OK	OK	OK	OK	OK	---	OK	
	MULTIPLE BIT SET	SETA	530	SETA(530) turns ON the specified number of consecutive bits.	OK	OK	OK	OK	OK	OK	OK	OK	88
		@SETA			OK	OK	OK	OK	OK	OK	OK	OK	
	MULTIPLE BIT RESET	RSTA	531	RSTA(531) turns OFF the specified number of consecutive bits.	OK	OK	OK	OK	OK	OK	OK	OK	88
		@RSTA			OK	OK	OK	OK	OK	OK	OK	OK	
	SINGLE BIT SET	SETB	532	SETB(532) turns ON the specified bit.	OK	OK	OK	OK	OK	OK	OK	---	90
		@SETB			OK	OK	OK	OK	OK	OK	OK	---	
		!SETB			OK	OK	OK	OK	OK	OK	---	---	
		!@SETB			OK	OK	OK	OK	OK	OK	---	---	

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Sequence Output Instructions	SINGLE BIT RESET	RSTB	533	RSTB(533) turns OFF the specified bit.	OK	OK	OK	OK	OK	OK	OK	---	90
		@RSTB			OK	OK	OK	OK	OK	OK	OK	---	
		!RSTB			OK	OK	OK	OK	OK	OK	---	---	
		!@RSTB			OK	OK	OK	OK	OK	OK	---	---	
	SINGLE BIT OUTPUT	OUTB	534	OUTB(534) outputs the status of the instruction's execution condition to the specified bit.	OK	OK	OK	OK	OK	OK	OK	---	92
		@OUTB			OK	OK	OK	OK	OK	OK	OK	---	
		!OUTB			OK	OK	OK	OK	OK	OK	---	---	
Sequence Control Instructions	END	END	001	Indicates the end of a program.	OK	OK	OK	OK	OK	OK	OK	OK	97
	NO OPERATION	NOP	000	This instruction has no function.	OK	OK	OK	OK	OK	OK	OK	OK	98
	INTERLOCK	IL	002	Interlocks all outputs between IL(002) and ILC(003) when the execution condition for IL(002) is OFF.	OK	OK	OK	OK	OK	OK	OK	OK	99
	INTERLOCK CLEAR	ILC	003	Indicates the end of the interlock range.	OK	OK	OK	OK	OK	OK	OK	OK	99
	MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	517	Interlocks all outputs between MILH(517) and MILC(519) when the execution condition for MILH(517) is OFF.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	103
	MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	518	Interlocks all outputs between MILR(518) and MILC(519) when the execution condition for MILR(518) is OFF.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	103
	MULTI-INTERLOCK CLEAR	MILC	519	Indicates the end of a multi-interlock range by means of an MILH or MILR instruction with the same interlock number.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	103
	JUMP	JMP	004	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.	OK	OK	OK	OK	OK	OK	OK	OK	112
	JUMP END	JME	005	Indicates the end position of a jump by JMP or CJP instruction.	OK	OK	OK	OK	OK	OK	OK	OK	112
	CONDITIONAL JUMP	CJP	510	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.	OK	OK	OK	OK	OK	OK	OK	OK	115
	CONDITIONAL JUMP NOT	CJPN	511	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.	OK	OK	OK	OK	OK	OK	OK	OK	115
	MULTIPLE JUMP	JMP0	515	When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000).	OK	OK	OK	OK	OK	OK	OK	OK	118
	MULTIPLE JUMP END	JME0	516		OK	OK	OK	OK	OK	OK	OK	OK	118
	---	FOR	512	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times.	OK	OK	OK	OK	OK	OK	OK	OK	121
	---	NEXT	513		OK	OK	OK	OK	OK	OK	OK	OK	121
	BREAK LOOP	BREAK	514	Programmed in a FOR-NEXT loop to cancel the execution of the loop for a given execution condition. The remaining instructions in the loop are processed as NOP(000) instructions.	OK	OK	OK	OK	OK	OK	OK	OK	124
Timer and Counter Instructions	HUNDRED-MS TIMER	TIM	---	TIM or TIMX(550) operates a decrementing timer with units of 0.1-s.	OK	OK	OK	OK	OK	OK	OK	OK	133
		TIMX	550		OK	OK	OK	OK	OK	OK	OK	---	133
	TEN-MS TIMER	TIMH	015	TIMH(015)/TIMHX(551) operates a decrementing timer with units of 10-ms.	OK	OK	OK	OK	OK	OK	OK	OK	137
		TIMHX	551		OK	OK	OK	OK	OK	OK	OK	---	137
	ONE-MS TIMER	TMHH	540	TMHH(540)/TMHHX(552) operates a decrementing timer with units of 1-ms.	OK	OK	OK	OK	OK	OK	OK	OK	141
		TMHHX	552		OK	OK	OK	OK	OK	OK	OK	---	141
	TENTH-MS TIMER	TIMU	541	TIMU(541)/TIMUX(556) operates a decrementing timer with units of 0.1-ms.	OK	OK	CJ1-H-R only	---	---	---	---	---	144
		TIMUX	556		OK	OK	CJ1-H-R only	---	---	---	---	---	144
	HUNDREDTH-MS TIMER	TMUH	544	TMUH(544)/TMUHX(557) operates a decrementing timer with units of 0.01-ms.	OK	OK	CJ1-H-R only	---	---	---	---	---	147
		TMUHX	557		OK	OK	CJ1-H-R only	---	---	---	---	---	147
	ACCUMULATIVE TIMER	TTIM	087	TTIM(087)/TTIMX(555) operates an incrementing timer with units of 0.1-s.	OK	OK	OK	OK	OK	OK	OK	OK	150
		TTIMX	555		OK	OK	OK	OK	OK	OK	OK	---	150

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Timer and Counter Instructions	LONG TIMER	TIML	542	TIML(542)/TIMLX(553) operates a decrementing timer with units of 0.1s.	OK	OK	OK	OK	OK	OK	OK	OK	153
		TIMLX	553		OK	OK	OK	OK	OK	OK	OK	---	153
	MULTI-OUTPUT TIMER	MTIM	543	MTIM(543)/MTIMX(554) operates a 0.1-s incrementing timer with eight independent SVs and Completion Flags.	OK	OK	OK	OK	OK	OK	OK	OK	156
		MTIMX	554		OK	OK	OK	OK	OK	OK	OK	---	156
	COUNTER	CNT	---	CNT/CNTX(546) operates a decrementing counter.	OK	OK	OK	OK	OK	OK	OK	OK	160
		CNTX	546		OK	OK	OK	OK	OK	OK	OK	---	160
	REVERSIBLE COUNTER	CNTR	012	---	OK	OK	OK	OK	OK	OK	OK	OK	163
		CNTRX	548		OK	OK	OK	OK	OK	OK	OK	---	163
	RESET TIMER/COUNTER	CNR/ @CNR	545	Resets the timers or counters within the specified range of timer or counter numbers.	OK	OK	OK	OK	OK	OK	OK	OK	166
		CNRX/ @CNRX	547		OK	OK	OK	OK	OK	OK	OK	---	166
	TIMER RESET	TRSET/ @TRSET	549	Resets the specified timer.	OK	OK	---	---	---	---	---	---	168
Comparison Instructions	Input Comparison Instructions	= , <> , < , <= , > , >=	300 to 328	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.	OK	OK	OK	OK	OK	OK	OK	OK	169
	Time Comparison Instructions	LD, AND, OR+=DT	341	Time comparison instructions compare two BCD time values and create an ON execution condition when the comparison condition is true.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	173
		LD, AND, OR+<>DT	342		OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	
		LD, AND, OR+<DT	343		OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	
		LD, AND, OR+<=DT	344		OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	
		LD, AND, OR+>DT	345		OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	
		LD, AND, OR+>=DT	346		OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	
	COMPARE	CMP	020	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.	OK	OK	OK	OK	OK	OK	OK	OK	177
		!CMP			OK	OK	OK	OK	OK	OK	---	OK	
	DOUBLE COMPARE	CMPL	060	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.	OK	OK	OK	OK	OK	OK	OK	OK	177
	SIGNED BINARY COMPARE	CPS	114	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.	OK	OK	OK	OK	OK	OK	OK	OK	180
		!CPS			OK	OK	OK	OK	OK	OK	---	OK	
	DOUBLE SIGNED BINARY COMPARE	CPSL	115	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.	OK	OK	OK	OK	OK	OK	OK	OK	180
	MULTIPLE COMPARE	MCMP	019	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.	OK	OK	OK	OK	OK	OK	OK	OK	183
		@MCMP			OK	OK	OK	OK	OK	OK	OK	OK	
	TABLE COMPARE	TCMP	085	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.	OK	OK	OK	OK	OK	OK	OK	OK	185
		@TCMP			OK	OK	OK	OK	OK	OK	OK	OK	
	BLOCK COMPARE	BCMP	068	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.	OK	OK	OK	OK	OK	OK	OK	OK	187
		@BCMP			OK	OK	OK	OK	OK	OK	OK	OK	
	EXPANDED BLOCK COMPARE	BCMP2	502	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.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	189

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Comparison Instructions	AREA RANGE COMPARE	ZCP	088	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.	OK	OK	OK	OK	OK	OK	OK	---	192
	DOUBLE AREA RANGE COMPARE	ZCPL	116	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.	OK	OK	OK	OK	OK	OK	OK	---	192
	SIGNED AREA RANGE COMPARE	ZCPS	117	Compares the 16-bit signed binary value in CD (word contents or constant) to the range defined by LL and UL and outputs the results to the Arithmetic Flags.	Ver. 1.3 or later	OK	---	---	---	---	---	---	196
	DOUBLE SIGNED AREA RANGE COMPARE	ZCPSL	118	Compares the 32-bit signed binary value in CD and CD+1 (word contents or constant) to the range defined by LL+1, LL and UL+1, UL and outputs the results to the Arithmetic Flags.	Ver. 1.3 or later	OK	---	---	---	---	---	---	196
Data Movement Instructions	MOVE	MOV	021	Transfers a word of data to the specified word.	OK	OK	OK	OK	OK	OK	OK	OK	199
		@MOV			OK	OK	OK	OK	OK	OK	OK	OK	
		!MOV			OK	OK	OK	OK	OK	OK	---	OK	
		!@MOV			OK	OK	OK	OK	OK	OK	---	OK	
	DOUBLE MOVE	MOVL/ @MOVL	498	Transfers two words of data to the specified words.	OK	OK	OK	OK	OK	OK	OK	OK	199
	MOVE NOT	MVN/ @MVN	022	Transfers the complement of a word of data to the specified word.	OK	OK	OK	OK	OK	OK	OK	OK	202
	DOUBLE MOVE NOT	MVNL/ @MVNL	499	Transfers the complement of two words of data to the specified words.	OK	OK	OK	OK	OK	OK	OK	OK	202
	MOVE BIT	MOVB/ @MOVB	082	Transfers the specified bit.	OK	OK	OK	OK	OK	OK	OK	OK	204
	MOVE DIGIT	MOVD/ @MOVD	083	Transfers the specified digit or digits. (Each digit is made up of 4 bits.)	OK	OK	OK	OK	OK	OK	OK	OK	206
	MULTIPLE BIT TRANSFER	XFRB/ @XFRB	062	Transfers the specified number of consecutive bits.	OK	OK	OK	OK	OK	OK	OK	OK	208
	BLOCK TRANSFER	XFER/ @XFER	070	Transfers the specified number of consecutive words.	OK	OK	OK	OK	OK	OK	OK	OK	210
	BLOCK SET	BSET/ @BSET	071	Copies the same word to a range of consecutive words.	OK	OK	OK	OK	OK	OK	OK	OK	212
	DATA EXCHANGE	XCHG/ @XCHG	073	Exchanges the contents of the two specified words.	OK	OK	OK	OK	OK	OK	OK	OK	214
	DOUBLE DATA EXCHANGE	XCGL/ @XCGL	562	Exchanges the contents of a pair of consecutive words with another pair of consecutive words.	OK	OK	OK	OK	OK	OK	OK	OK	214
	SINGLE WORD DISTRIBUTE	DIST/ @DIST	080	Transfers the source word to a destination word calculated by adding an offset value to the base address.	OK	OK	OK	OK	OK	OK	OK	OK	216
	DATA COLLECT	COLL/ @COLL	081	Transfers the source word (calculated by adding an offset value to the base address) to the destination word.	OK	OK	OK	OK	OK	OK	OK	OK	218
	MOVE TO REGISTER	MOVR/ @MOVR	560	Sets the PLC memory address of the specified word, bit, or timer/counter Completion Flag in the specified Index Register.	OK	OK	OK	OK	OK	OK	OK	OK	220
	MOVE TIMER/COUNTER PV TO REGISTER	MOVRW/ @MOVRW	561	Sets the PLC memory address of the specified timer or counter's PV in the specified Index Register.	OK	OK	OK	OK	OK	OK	OK	OK	220
Data Shift Instruction	SHIFT REGISTER	SFT	010	Operates a shift register.	OK	OK	OK	OK	OK	OK	OK	OK	223
	REVERSIBLE SHIFT REGISTER	SFTR/ @SFTR	084	Creates a shift register that shifts data to either the right or the left.	OK	OK	OK	OK	OK	OK	OK	OK	225
	ASYNCHRONOUS SHIFT REGISTER	ASFT/ @ASFT	017	Shifts all non-zero word data within the specified word range either towards St or toward E, replacing 0000Hex word data.	OK	OK	OK	OK	OK	OK	OK	OK	227
	WORD SHIFT	WSFT/ @WSFT	016	Shifts data between St and E in word units.	OK	OK	OK	OK	OK	OK	OK	OK	229
	ARITHMETIC SHIFT LEFT	ASL/ @ASL	025	Shifts the contents of Wd one bit to the left.	OK	OK	OK	OK	OK	OK	OK	OK	231
	DOUBLE SHIFT LEFT	ASLL/ @ASLL	570	Shifts the contents of Wd and Wd +1 one bit to the left.	OK	OK	OK	OK	OK	OK	OK	OK	231

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Data Shift Instruction	ARITHMETIC SHIFT RIGHT	ASR/@ASR	026	Shifts the contents of Wd one bit to the right.	OK	OK	OK	OK	OK	OK	OK	OK	233
	DOUBLE SHIFT RIGHT	ASRL/@ASRL	571	Shifts the contents of Wd and Wd +1 one bit to the right.	OK	OK	OK	OK	OK	OK	OK	OK	233
	ROTATE LEFT	ROL/@ROL	027	Shifts all Wd bits one bit to the left including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	235
	DOUBLE ROTATE LEFT	ROLL/@ROLL	572	Shifts all Wd and Wd +1 bits one bit to the left including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	235
	ROTATE LEFT WITHOUT CARRY	RLNC/@RLNC	574	Shifts all Wd bits one bit to the left not including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	237
	DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL/@RLNL	576	Shifts all Wd and Wd +1 bits one bit to the left not including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	237
	ROTATE RIGHT	ROR/@ROR	028	Shifts all Wd bits one bit to the right including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	239
	DOUBLE ROTATE RIGHT	RORL/@RORL	573	Shifts all Wd and Wd +1 bits one bit to the right including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	239
	ROTATE RIGHT WITHOUT CARRY	RRNC/@RRNC	575	Shifts all Wd bits one bit to the right not including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	241
	DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL/@RRNL	577	Shifts all Wd and Wd +1 bits one bit to the right not including the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	241
	ONE DIGIT SHIFT LEFT	SLD/@SLD	074	Shifts data by one digit (4 bits) to the left.	OK	OK	OK	OK	OK	OK	OK	OK	243
	ONE DIGIT SHIFT RIGHT	SRD/@SRD	075	Shifts data by one digit (4 bits) to the right.	OK	OK	OK	OK	OK	OK	OK	OK	243
	SHIFT N-BIT DATA LEFT	NSFL/@NSFL	578	Shifts the specified number of bits to the left.	OK	OK	OK	OK	OK	OK	OK	OK	245
	SHIFT N-BIT DATA RIGHT	NSFR/@NSFR	579	Shifts the specified number of bits to the right.	OK	OK	OK	OK	OK	OK	OK	OK	245
	SHIFT N-BITS LEFT	NASL/@NASL	580	Shifts the specified 16 bits of word data to the left by the specified number of bits.	OK	OK	OK	OK	OK	OK	OK	OK	248
	DOUBLE SHIFT N-BITS LEFT	NSLL/@NSLL	582	Shifts the specified 32 bits of word data to the left by the specified number of bits.	OK	OK	OK	OK	OK	OK	OK	OK	248
	SHIFT N-BITS RIGHT	NASR/@NASR	581	Shifts the specified 16 bits of word data to the right by the specified number of bits.	OK	OK	OK	OK	OK	OK	OK	OK	251
	DOUBLE SHIFT N-BITS RIGHT	NSRL/@NSRL	583	Shifts the specified 32 bits of word data to the right by the specified number of bits.	OK	OK	OK	OK	OK	OK	OK	OK	251
Increment/ Decrement Instructions	INCREMENT BINARY	++/@++	590	Increments the 4-digit hexadecimal content of the specified word by 1.	OK	OK	OK	OK	OK	OK	OK	OK	254
	DOUBLE INCREMENT BINARY	++L/@++L	591	Increments the 8-digit hexadecimal content of the specified words by 1.	OK	OK	OK	OK	OK	OK	OK	OK	254
	DECREMENT BINARY	--/@--	592	Decrements the 4-digit hexadecimal content of the specified word by 1.	OK	OK	OK	OK	OK	OK	OK	OK	257
	DOUBLE DECREMENT BINARY	--L/@--L	593	Decrements the 8-digit hexadecimal content of the specified words by 1.	OK	OK	OK	OK	OK	OK	OK	OK	257
	INCREMENT BCD	++B/@++B	594	Increments the 4-digit BCD content of the specified word by 1.	OK	OK	OK	OK	OK	OK	OK	OK	260
	DOUBLE INCREMENT BCD	++BL/@++BL	595	Increments the 8-digit BCD content of the specified words by 1.	OK	OK	OK	OK	OK	OK	OK	OK	260
	DECREMENT BCD	--B/@--B	596	Decrements the 4-digit BCD content of the specified word by 1.	OK	OK	OK	OK	OK	OK	OK	OK	263
	DOUBLE DECREMENT BCD	--BL/@--BL	597	Decrements the 8-digit BCD content of the specified words by 1.	OK	OK	OK	OK	OK	OK	OK	OK	263

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Symbol Math Instructions	SIGNED BINARY ADD WITHOUT CARRY	+/ @+	400	Adds 4-digit (single-word) hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	266
	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L/ @+L	401	Adds 8-digit (double-word) hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	266
	SIGNED BINARY ADD WITH CARRY	+C/ @+C	402	Adds 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	268
	DOUBLE SIGNED BINARY ADD WITH CARRY	+CL/ @+CL	403	Adds 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	268
	BCD ADD WITHOUT CARRY	+B/ @+B	404	Adds 4-digit (single-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	270
	DOUBLE BCD ADD WITHOUT CARRY	+BL/ @+BL	405	Adds 8-digit (double-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	270
	BCD ADD WITH CARRY	+BC/@+BC	406	Adds 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	272
	DOUBLE BCD ADD WITH CARRY	+BCL/@+BCL	407	Adds 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	272
	SIGNED BINARY SUBTRACT WITHOUT CARRY	-/ @-	410	Subtracts 4-digit (single-word) hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	274
	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	-L/ @-L	411	Subtracts 8-digit (double-word) hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	274
	SIGNED BINARY SUBTRACT WITH CARRY	-C/ @-C	412	Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	278
	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	-CL/ @CL	413	Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	278
	BCD SUBTRACT WITHOUT CARRY	-B/ @-B	414	Subtracts 4-digit (single-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	281
	DOUBLE BCD SUBTRACT WITHOUT CARRY	-BL/ @-BL	415	Subtracts 8-digit (double-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	281
	BCD SUBTRACT WITH CARRY	-BC/ @-BC	416	Subtracts 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	284
	DOUBLE BCD SUBTRACT WITH CARRY	-BCL/ @-BCL	417	Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	284
	SIGNED BINARY MULTIPLY	*/ @*	420	Multiplies 4-digit signed hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	286
	DOUBLE SIGNED BINARY MULTIPLY	*L/ @*L	421	Multiplies 8-digit signed hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	286
	UNSIGNED BINARY MULTIPLY	*U/ @*U	422	Multiplies 4-digit unsigned hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	288
	DOUBLE UNSIGNED BINARY MULTIPLY	*UL/ @*UL	423	Multiplies 8-digit unsigned hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	288
	BCD MULTIPLY	*B/ @*B	424	Multiplies 4-digit (single-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	290
	DOUBLE BCD MULTIPLY	*BL/ @*BL	425	Multiplies 8-digit (double-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	290
	SIGNED BINARY DIVIDE	/ @/	430	Divides 4-digit (single-word) signed hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	292

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Symbol Math Instructions	DOUBLE SIGNED BINARY DIVIDE	/L @/L	431	Divides 8-digit (double-word) signed hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	292
	UNSIGNED BINARY DIVIDE	/U @/U	432	Divides 4-digit (single-word) unsigned hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	294
	DOUBLE UNSIGNED BINARY DIVIDE	/UL @/UL	433	Divides 8-digit (double-word) unsigned hexadecimal data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	294
	BCD DIVIDE	/B @/B	434	Divides 4-digit (single-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	296
	DOUBLE BCD DIVIDE	/BL @/BL	435	Divides 8-digit (double-word) BCD data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	296
Conversion Instructions	BCD TO BINARY	BIN/ @BIN	023	Converts BCD data to binary data.	OK	OK	OK	OK	OK	OK	OK	OK	298
	DOUBLE BCD TO DOUBLE BINARY	BINL/ @BINL	058	Converts 8-digit BCD data to 8-digit hexadecimal (32-bit binary) data.	OK	OK	OK	OK	OK	OK	OK	OK	298
	BINARY TO BCD	BCD/ @BCD	024	Converts a word of binary data to a word of BCD data.	OK	OK	OK	OK	OK	OK	OK	OK	300
	DOUBLE BINARY TO DOUBLE BCD	BCDL/ @BCDL	059	Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data.	OK	OK	OK	OK	OK	OK	OK	OK	300
	2'S COMPLEMENT	NEG/ @NEG	160	Calculates the 2's complement of a word of hexadecimal data.	OK	OK	OK	OK	OK	OK	OK	OK	303
	DOUBLE 2'S COMPLEMENT	NEGL/ @NEGL	161	Calculates the 2's complement of two words of hexadecimal data.	OK	OK	OK	OK	OK	OK	OK	OK	303
	16-BIT TO 32-BIT SIGNED BINARY	SIGN/ @SIGN	600	Expands a 16-bit signed binary value to its 32-bit equivalent.	OK	OK	OK	OK	OK	OK	OK	OK	305
	DATA DECODER	MLPX/ @MLPX	076	Reads the numerical value in the specified digit (or byte) in the source word with 4-to 16 conversion (or 8-to-256 conversion), turns ON the corresponding bit in the result word, and turns OFF all other bits in the result word.	OK	OK	OK	OK	OK	OK	OK	OK	307
	DATA ENCODER	DMPX/ @DMPX	077	Finds the location of the first or last ON bit within the source word with 16-to-4 conversion (or 256-to-8 conversion), and writes that value to the specified digit (or byte) in the result word.	OK	OK	OK	OK	OK	OK	OK	OK	312
	ASCII CONVERT	ASC/ @ASC	086	Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents.	OK	OK	OK	OK	OK	OK	OK	OK	317
	ASCII TO HEX	HEX/ @HEX	162	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.	OK	OK	OK	OK	OK	OK	OK	OK	321
	COLUMN TO LINE	LINE/ @LINE	063	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.	OK	OK	OK	OK	OK	OK	OK	OK	326
	COLUMN TO LINE	COLM/ @COLM	064	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.	OK	OK	OK	OK	OK	OK	OK	OK	328
	SIGNED BCD TO BINARY	BINS/ @BINS	470	Converts one word of signed BCD data to one word of signed binary data.	OK	OK	OK	OK	OK	OK	OK	OK	330
	DOUBLE SIGNED BCD TO BINARY	BISL/ @BISL	472	Converts double signed BCD data to double signed binary data.	OK	OK	OK	OK	OK	OK	OK	OK	330
	SIGNED BINARY TO BCD	BCDS/ @BCDS	471	Converts one word of signed binary data to one word of signed BCD data.	OK	OK	OK	OK	OK	OK	OK	OK	335
	DOUBLE SIGNED BINARY TO BCD	BDSL/ @BDSL	473	Converts double signed binary data to double signed BCD data.	OK	OK	OK	OK	OK	OK	OK	OK	335
	GRAY CODE CONVERT	GRY	474	Converts the gray binary code in a specified word to standard binary data, BCD data, or an angle at the specified resolution.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	339
	GRAY CODE TO BINARY CONVERT	GRAY_BIN/ @GRAY_BIN	478	Converts the specified word of gray code to one word of binary data.	OK	OK	---	---	---	---	---	---	344

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Conversion Instructions	DOUBLE GRAY CODE TO BINARY CONVERT	GRAY_BINL/ @GRAY_BINL	479	Converts the specified two words of gray code to two words of binary data.	OK	OK	---	---	---	---	---	---	344
	BINARY TO GRAY CODE CONVERT	BIN_GRAY/ @BIN_GRAY	480	Converts the specified word of binary data to one word of gray code.	OK	OK	---	---	---	---	---	---	346
	DOUBLE BINARY TO GRAY CODE CONVERT	BIN_GRAYL/ @BIN_GRAYL	481	Converts the specified two words of binary data to two words of gray code.	OK	OK	---	---	---	---	---	---	346
	FOUR-DIGIT NUMBER TO ASCII	STR4	601	Converts a 4-digit hexadecimal number (#0000 to #FFFF) to ASCII data (4 characters).	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	348
	EIGHT-DIGIT NUMBER TO ASCII	STR8	602	Converts an 8-digit hexadecimal number (#0000 0000 to #FFFF FFFF) to ASCII data (8 characters).	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	348
	SIXTEEN-DIGIT NUMBER TO ASCII	STR16	603	Converts a 16-digit hexadecimal number (#0000 0000 0000 0000 to #FFFF FFFF FFFF FFFF) to ASCII data (16 characters).	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	348
	FOUR-DIGIT NUMBER TO ASCII	NUM4	604	Converts 4 characters of ASCII data to a 4-digit hexadecimal number.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	351
	EIGHT-DIGIT NUMBER TO ASCII	NUM8	605	Converts 8 characters of ASCII data to an 8-digit hexadecimal number.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	351
	SIXTEEN-DIGIT NUMBER TO ASCII	NUM16	606	Converts 16 characters of ASCII data to an 16-digit hexadecimal number.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	351
Logic Instructions	LOGICAL AND	ANDW/ @ANDW	034	Takes the logical AND of corresponding bits in single words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	354
	DOUBLE LOGICAL AND	ANDL/ @ANDL	610	Takes the logical AND of corresponding bits in double words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	354
	LOGICAL OR	ORW/ @ORW	035	Takes the logical OR of corresponding bits in single words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	356
	DOUBLE LOGICAL OR	ORWL/ @ORWL	611	Takes the logical OR of corresponding bits in double words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	356
	EXCLUSIVE OR	XORW/ @XORW	036	Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	358
	DOUBLE EXCLUSIVE OR	XORL/ @XORL	612	Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	358
	EXCLUSIVE NOR	XNRW/ @XNRW	037	Takes the logical exclusive NOR of corresponding single words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	360
	DOUBLE EXCLUSIVE NOR	XNRL/ @XNRL	613	Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants.	OK	OK	OK	OK	OK	OK	OK	OK	360
	COMPLEMENT	COM/ @COM	029	Turns OFF all ON bits and turns ON all OFF bits in Wd.	OK	OK	OK	OK	OK	OK	OK	OK	362
	DOUBLE COMPLEMENT	COML/ @COML	614	Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.	OK	OK	OK	OK	OK	OK	OK	OK	362
Special Math Instructions	BINARY ROOT	ROTB/ @ROTB	620	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.	OK	OK	OK	OK	OK	OK	OK	OK	364
	BCD SQUARE ROOT	ROOT/ @ROOT	072	Computes the square root of an 8-digit BCD number and outputs the integer portion of the result to the specified result word.	OK	OK	OK	OK	OK	OK	OK	OK	366
	ARITHMETIC PROCESS	APR/ @APR	069	Calculates the sine, cosine, or a linear extrapolation of the source data.	OK	OK	OK	OK	OK	OK	OK	OK	369
	FLOATING POINT DIVIDE (BCD)	FDIV/ @FDIV	079	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).	OK	OK	OK	OK	OK	OK	OK	OK	378
	BIT COUNTER	BCNT/ @BCNT	067	Counts the total number of ON bits in the specified word(s).	OK	OK	OK	OK	OK	OK	OK	OK	381

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Floating-point Math Instructions	FLOATING TO 16-BIT	FIX/ @FIX	450	Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word.	OK	OK	OK	OK	OK	OK	OK	OK	388
	FLOATING TO 32-BIT	FIXL/ @FIXL	451	Converts a 32-bit floating-point value to 32-bit signed binary data and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	388
	16-BIT TO FLOATING	FLT/ @FLT	452	Converts a 16-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	390
	32-BIT TO FLOATING	FLTL/ @FLTL	453	Converts a 32-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	390
	FLOATING-POINT ADD	+F/ @+F	454	Adds two 32-bit floating-point numbers and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	392
	FLOATING-POINT SUBTRACT	-F/ @-F	455	Subtracts one 32-bit floating-point number from another and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	392
	FLOATING-POINT MULTIPLY	*F/ @*F	456	Multiplies two 32-bit floating-point numbers and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	392
	FLOATING-POINT DIVIDE	/F @/F	457	Divides one 32-bit floating-point number by another and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	392
	DEGREES TO RADIANS	RAD/ @RAD	458	Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	396
	RADIANS TO DEGREES	DEG/ @DEG	459	Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	398
	SINE	SIN/ @SIN	460	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	400
	COSINE	COS/ @COS	461	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	400
	TANGENT	TAN/ @TAN	462	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	400
	HIGH-SPEED SINE	SINQ/ @SINQ	475	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.	OK	OK	CJ1-H-R only	---	---	---	---	---	403
	HIGH-SPEED COSINE	COSQ/ @COSQ	476	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.	OK	OK	CJ1-H-R only	---	---	---	---	---	403
	HIGH-SPEED TANGENT	TANQ/ @TANQ	477	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.	OK	OK	CJ1-H-R only	---	---	---	---	---	403
	ARC SINE	ASIN/ @ASIN	463	Calculates the arc sine of a 32-bit floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	407
	ARC COSINE	ACOS/ @ACOS	464	Calculates the arc cosine of a 32-bit floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	407
	ARC TANGENT	ATAN/ @ATAN	465	Calculates the arc tangent of a 32-bit floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	407
	SQUARE ROOT	SQRT/ @SQRT	466	Calculates the square root of a 32-bit floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	410
	EXPONENT	EXP/ @EXP	467	Calculates the natural (base e) exponential of a 32-bit floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	412
	LOGARITHM	LOG/ @LOG	468	Calculates the natural (base e) logarithm of a 32-bit floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	OK	414

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Floating-point Math Instructions	EXPONENTIAL POWER	PWR/ @PWR	840	Raises a 32-bit floating-point number to the power of another 32-bit floating-point number.	OK	OK	OK	OK	OK	OK	OK	OK	416
	Single-precision Floating-point Comparison	LD, AND, OR+=F	329	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.	OK	OK	OK	OK	OK	OK	OK	---	418
		LD, AND, OR+<>F	330		OK	OK	OK	OK	OK	OK	OK	---	418
		LD, AND, OR+<F	331		OK	OK	OK	OK	OK	OK	OK	---	418
		LD, AND, OR+<=F	332		OK	OK	OK	OK	OK	OK	OK	---	418
		LD, AND, OR+>F	333		OK	OK	OK	OK	OK	OK	OK	---	418
		LD, AND, OR+>=F	334		OK	OK	OK	OK	OK	OK	OK	---	418
	FLOATING-POINT TO ASCII	FSTR/ @FSTR	448	Expresses a 32-bit floating-point value (IEEE754-format) in standard decimal notation or scientific notation and converts that value to ASCII text.	OK	OK	OK	OK	OK	OK	OK	---	421
	ASCII TO FLOATING-POINT	FVAL/ @FVAL	449	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.	OK	OK	OK	OK	OK	OK	OK	---	426
	MOVE FLOATING-POINT (SINGLE)	MOVF/ @MOVF	469	Transfers the specified 32-bit floating-point number to the destination words.	OK	OK	CJ1-H-R only	---	---	---	---	---	430
Double-precision Floating-point Instructions	DOUBLE FLOATING TO 16-BIT	FIXD/ @FIXD	841	Converts a double-precision (64-bit) floating-point value to 16-bit signed binary data and places the result in the specified result word.	OK	OK	OK	OK	OK	OK	OK	---	437
	DOUBLE FLOATING TO 32-BIT	FIXLD/ @FIXLD	842	Converts a double-precision (64-bit) floating-point value to 32-bit signed binary data and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	437
	16-BIT TO DOUBLE FLOATING	DBL/ @DBL	843	Converts a 16-bit signed binary value to double-precision (64-bit) floating-point data and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	439
	32-BIT TO DOUBLE FLOATING	DBLL/ @DBLL	844	Converts a double-precision (64-bit) floating-point value to 32-bit signed binary data and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	439
	DOUBLE FLOATING-POINT ADD	+D/ @+D	845	Adds two double-precision (64-bit) floating-point numbers and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	441
	DOUBLE FLOATING-POINT SUBTRACT	-D/ @-D	846	Subtracts one double-precision (64-bit) floating-point number from another and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	441
	DOUBLE FLOATING-POINT MULTIPLY	*D/ @*D	847	Multiplies two double-precision (64-bit) floating-point numbers and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	441
	DOUBLE FLOATING-POINT DIVIDE	/D @/D	848	Divides one double-precision (64-bit) floating-point number by another and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	441
	DOUBLE DEGREES TO RADIANS	RADD/ @RADD	849	Converts a double-precision (64-bit) floating-point number from degrees to radians and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	444
	DOUBLE RADIANS TO DEGREES	DEGD/ @DEGD	850	Converts a double-precision (64-bit) floating-point number from radians to degrees and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	446
	DOUBLE SINE	SIND/ @SIND	851	Calculates the sine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	448
	DOUBLE COSINE	COSD/ @COSD	852	Calculates the cosine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	448

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Double-precision Floating-point Instructions	DOUBLE TANGENT	TAND/ @TAND	853	Calculates the tangent of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	448
	DOUBLE ARC SINE	ASIND/ @ASIND	854	Calculates the arc sine of a double-precision (64-bit) floating-point number and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	451
	DOUBLE ARC COSINE	ACOSD/ @ACOSD	855	Calculates the arc cosine of a double-precision (64-bit) floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	451
	DOUBLE ARC TANGENT	ATAND/ @ATAND	856	Calculates the arc tangent of a double-precision (64-bit) floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	451
	DOUBLE SQUARE ROOT	SQRTD/ @SQRTD	857	Calculates the square root of a double-precision (64-bit) floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	454
	DOUBLE EXPONENT	EXPD/ @EXPD	858	Calculates the natural (base e) exponential of a double-precision (64-bit) floating-point number and places the result in the specified result words.	OK	OK	OK	OK	OK	OK	OK	---	456
	DOUBLE LOGARITHM	LOGD/ @LOGD	859	Calculates the natural (base e) logarithm of a double-precision (64-bit) floating-point number and places the result in the specified destination words.	OK	OK	OK	OK	OK	OK	OK	---	458
	DOUBLE EXPONENTIAL POWER	PWRD/ @PWRD	860	Raises a double-precision (64-bit) floating-point number to the power of another double-precision (64-bit) floating-point number.	OK	OK	OK	OK	OK	OK	OK	---	460
	Double-precision Floating-point Input Comparison	LD, AND, OR+=D	335	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.	OK	OK	OK	OK	OK	OK	OK	---	462
		LD, AND, OR+<>D	336		OK	OK	OK	OK	OK	OK	OK	---	462
		LD, AND, OR+<D	337		OK	OK	OK	OK	OK	OK	OK	---	462
		LD, AND, OR+<=D	338		OK	OK	OK	OK	OK	OK	OK	---	462
		LD, AND, OR+>D	339		OK	OK	OK	OK	OK	OK	OK	---	462
		LD, AND, OR+>=D	340		OK	OK	OK	OK	OK	OK	OK	---	462
Table Data Processing Instructions	SET STACK	SSET/ @SSET	630	Defines a stack of the specified length beginning at the specified word.	OK	OK	OK	OK	OK	OK	OK	OK	470
	PUSH ONTO STACK	PUSH/ @PUSH	632	Writes one word of data to the specified stack.	OK	OK	OK	OK	OK	OK	OK	OK	473
	LAST IN FIRST OUT	LIFO/ @LIFO	634	Reads the last word of data written to the specified stack (the newest data in the stack).	OK	OK	OK	OK	OK	OK	OK	OK	475
	FIRST IN FIRST OUT	FIFO/ @FIFO	633	Reads the first word of data written to the specified stack (the oldest data in the stack).	OK	OK	OK	OK	OK	OK	OK	OK	475
	STACK SIZE READ	SNUM/ @SNUM	638	Counts the amount of stack data (number of words) in the specified stack.	OK	OK	OK	OK	OK	OK	OK	---	479
	STACK DATA READ	SREAD/ @SREAD	639	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).	OK	OK	OK	OK	OK	OK	OK	---	481
	STACK DATA OVERWRITE	SWRIT/ @SWRIT	640	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).	OK	OK	OK	OK	OK	OK	OK	---	484
	STACK DATA INSERT	SINS/ @SINS	641	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).	OK	OK	OK	OK	OK	OK	OK	---	487

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Table Data Processing Instructions	STACK DATA DELETE	SDEL/ @SDEL	642	Deletes the data element at the specified location in the stack, outputs that data to the specified destination word, and shifts the remaining 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).	OK	OK	OK	OK	OK	OK	OK	---	490
	DIMENSION RECORD TABLE	DIM/ @DIM	631	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.	OK	OK	OK	OK	OK	OK	OK	OK	493
	SET RECORD LOCATION	SETR/ @SETR	635	Writes the location of the specified record (the PLC memory address of the beginning of the record) in the specified Index Register.	OK	OK	OK	OK	OK	OK	OK	OK	495
	GET RECORD NUMBER	GETR/ @GETR	636	Returns the record number of the record at the PLC memory address contained in the specified Index Register.	OK	OK	OK	OK	OK	OK	OK	OK	497
	DATA SEARCH	SRCH/ @SRCH	181	Searches for a word of data within a range of words.	OK	OK	OK	OK	OK	OK	OK	OK	499
	SWAP BYTES	SWAP/ @SWAP	637	Switches the leftmost and rightmost bytes in all of the words in the range.	OK	OK	OK	OK	OK	OK	OK	OK	502
	FIND MAXIMUM	MAX/ @MAX	182	Finds the maximum value in the range.	OK	OK	OK	OK	OK	OK	OK	OK	504
	FIND MINIMUM	MIN/ @MIN	183	Finds the minimum value in the range.	OK	OK	OK	OK	OK	OK	OK	OK	504
	DOUBLE FIND MAXIMUM	MAXL/ @MAXL	174	Treats the specified number of data items as double word table data and outputs the maximum value in the table.	OK	OK	---	---	---	---	---	---	508
	FIND MAXIMUM FLOATING	MAXF/ @MAXF	176	Treats the specified number of data items as a table of single-precision floating-point data and outputs the maximum value in the table.	OK	OK	---	---	---	---	---	---	511
	FIND DOUBLE MAXIMUM FLOATING	MAXD/ @MAXD	178	Treats the specified number of data items as a table of double-precision floating-point data and outputs the maximum value in the table.	OK	OK	---	---	---	---	---	---	513
	DOUBLE FIND MINIMUM	MINL/ @MINL	175	Treats the specified number of data items as double word table data and outputs the minimum value in the table.	OK	OK	---	---	---	---	---	---	515
	FIND MINIMUM FLOATING	MINF/ @MINF	177	Treats the specified number of data items as a table of single-precision floating-point data and outputs the minimum value in the table.	OK	OK	---	---	---	---	---	---	517
	FIND DOUBLE MINIMUM FLOATING	MIND/ @MIND	179	Treats the specified number of data items as a table of double-precision floating-point data and outputs the minimum value in the table.	OK	OK	---	---	---	---	---	---	519
	SUM	SUM/ @SUM	184	Adds the bytes or words in the range and outputs the result to two words.	OK	OK	OK	OK	OK	OK	OK	OK	521
	FRAME CHECKSUM	FCS/ @FCS	180	Calculates the FCS value for the specified range and outputs the result in ASCII.	OK	OK	OK	OK	OK	OK	OK	OK	524
Tracking Instructions	Unsigned One-word Record Search Instructions	RSRCH < , @RSRCH <	360	An Unsigned One-word Record Search Instruction searches the specified table of records for a record with the specific data (1 word) in the specified location. When a record matching the specified condition is found, its record number and data are output.	OK	OK	---	---	---	---	---	---	534
		RSRCH <= , @RSRCH <=	361		OK	OK	---	---	---	---	---	---	534
		RSRCH = , @RSRCH =	362		OK	OK	---	---	---	---	---	---	534
		RSRCH >= , @RSRCH >=	363		OK	OK	---	---	---	---	---	---	534
		RSRCH > , @RSRCH >	364		OK	OK	---	---	---	---	---	---	534

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Tracking Instructions	Unsigned Two-word Record Search Instructions	RSRCH2 < , @RSRCH2 <	370	An Unsigned Two-word Record Search Instruction searches the specified table of records for a record with the specific data (2 words) in the specified location. When a record matching the specified condition is found, its record number and data are output.	OK	OK	---	---	---	---	---	---	540
		RSRCH2 <= , @RSRCH2 <=	371		OK	OK	---	---	---	---	---	---	540
		RSRCH2 = , @RSRCH2 =	372		OK	OK	---	---	---	---	---	---	540
		RSRCH2 >= , @RSRCH2 >=	373		OK	OK	---	---	---	---	---	---	540
		RSRCH2 > , @RSRCH2 >	374		OK	OK	---	---	---	---	---	---	540
	Unsigned Four-word Record Search Instructions	RSRCH4 < , @RSRCH4 <	380	An Unsigned Four-word Record Search Instruction searches the specified table of records for a record with the specific data (4 words) in the specified location. When a record matching the specified condition is found, its record number and data are output.	OK	OK	---	---	---	---	---	---	543
		RSRCH4 <= , @RSRCH4 <=	381		OK	OK	---	---	---	---	---	---	543
		RSRCH4 = , @RSRCH4 =	382		OK	OK	---	---	---	---	---	---	543
		RSRCH4 >= , @RSRCH4 >=	383		OK	OK	---	---	---	---	---	---	543
		RSRCH4 > , @RSRCH4 >	384		OK	OK	---	---	---	---	---	---	543
	UNSIGNED ONE-WORD RECORD SORT	RSORT/ @RSORT	203	Sorts the records in the specified table according to the data (1 word) at the specified position in the records.	OK	OK	---	---	---	---	---	---	546
	UNSIGNED TWO-WORD RECORD SORT	RSORT2/ @RSORT2	204	Sorts the records in the specified table according to the data (2 words) at the specified position in the records.	OK	OK	---	---	---	---	---	---	550
	UNSIGNED FOUR-WORD RECORD SORT	RSORT4/ @RSORT4	205	Sorts the records in the specified table according to the data (4 words) at the specified position in the records.	OK	OK	---	---	---	---	---	---	553
Data Control Instructions	PID CONTROL	PID	190	Executes PID control according to the specified parameters.	OK	OK	OK	OK	OK	OK	OK	OK	556
	PID CONTROL WITH AUTO-TUNING	PIDAT	191	Executes PID control according to the specified parameters. The PID constants can be autotuned.	OK	OK	OK	OK	OK	OK	OK	---	567
	Controls output data according to whether or not input data is within upper and lower limits.	LMT/ @LMT	680	Controls output data according to whether or not input data is within upper and lower limits.	OK	OK	OK	OK	OK	OK	OK	OK	575
	DEAD BAND CONTROL	BAND/ @BAND	681	Controls output data according to whether or not input data is within the lower and upper limits of the range (dead band range.)	OK	OK	OK	OK	OK	OK	OK	OK	577
	DEAD ZONE CONTROL	ZONE/ @ZONE	682	Adds the specified bias to input data and outputs the result.	OK	OK	OK	OK	OK	OK	OK	OK	580
	TIME-PROPORTIONAL OUTPUT	TPO	685	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.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	582
	SCALING	SCL/ @SCL	194	Converts unsigned binary data into unsigned BCD data according to the specified linear function.	OK	OK	OK	OK	OK	OK	OK	OK	589
	SCALING 2	SCL2/ @SCL2	486	Converts signed binary data into signed BCD data according to the specified linear function. An offset can be input in defining the linear function.	OK	OK	OK	OK	OK	OK	OK	OK	593
	SCALING 3	SCL3/ @SCL3	487	Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function.	OK	OK	OK	OK	OK	OK	OK	OK	597
	AVERAGE	AVG	195	Calculates the average value of an input word for the specified number of cycles.	OK	OK	OK	OK	OK	OK	OK	OK	600

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Subroutines	SUBROUTINE CALL	SBS/ @SBS	091	Calls the subroutine with the specified subroutine number and executes that program.	OK	OK	OK	OK	OK	OK	OK	OK	604
	MACRO	MCRO/ @MCRO	099	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.	OK	OK	OK	OK	OK	OK	OK	OK	610
	SUBROUTINE ENTRY	SBN	092	Indicates the beginning of the subroutine program with the specified subroutine number.	OK	OK	OK	OK	OK	OK	OK	OK	613
	SUBROUTINE RETURN	RET	093	Indicates the end of a subroutine program.	OK	OK	OK	OK	OK	OK	OK	OK	613
	GLOBAL SUBROUTINE CALL	GSBS/ @GSBS	750	Calls the global subroutine with the specified subroutine number and executes that program.	OK	OK	OK	OK	OK	OK	OK	---	616
	GLOBAL SUBROUTINE ENTRY	GSBN	751	Indicates the beginning of the global subroutine program with the specified subroutine number.	OK	OK	OK	OK	OK	OK	OK	---	622
	GLOBAL SUBROUTINE RETURN	GRET	752	Indicates the end of a subroutine program.	OK	OK	OK	OK	OK	OK	OK	---	622
Interrupt Control Instructions	SET INTERRUPT MASK	MSKS/ @MSKS	690	Controls interrupts.	OK	OK	OK	OK	OK	OK	---	OK	628
	READ INTERRUPT MASK	MSKR/ @MSKR	692	Reads the current interrupt control settings that were set with MSKS(690).	OK	OK	OK	OK	OK	OK	---	OK	634
	CLEAR INTERRUPT	CLI/ @CLI	691	Clears/retains recorded interrupt inputs, sets the time to the first scheduled interrupt for scheduled interrupt tasks.	OK	OK	OK	OK	OK	OK	---	OK	639
	DISABLE INTERRUPTS	DI/ @DI	693	Disables execution of all interrupt tasks except the power OFF interrupt task for part of a program.	OK	OK	OK	OK	OK	OK	OK	OK	643
	ENABLE INTERRUPTS	EI	694	Enables execution of all interrupt tasks that were disabled with DI(693).	OK	OK	OK	OK	OK	OK	OK	OK	645
High-speed Counter/ Pulse Output Instructions	MODE CONTROL	INI/ @INI	880	INI(880) is used to start and stop comparison for a comparison table, to change the present value (PV) of a high-speed counter, to change the PV of an input interrupt in counter mode, to change the maximum value of the ring counter (CJ2M only), to change the PV of a pulse output (e.g., to 0 to establish the origin), to stop pulse output, or to change the settings for origin searches/returns (CJ2M only).	---	Ver. 2.0 or later	---	---	OK	---	---	---	647
	HIGH-SPEED COUNTER PV READ	PRV/ @PRV	881	PRV(881) reads the High-speed counter PV and pulse output PV and interrupt input PV in counter mode.	---	Ver. 2.0 or later	---	---	OK	---	---	---	653
	COUNTER FREQUENCY CONVERT	PRV2	883	Reads the pulse frequency input from a high-speed counter and either converts the frequency to a rotational speed (number of revolutions) or converts the counter PV to the total number of revolutions. The result is output to the destination words as 8-digit hexadecimal. The frequency measurement function can be used with high-speed counter 0 only.	---	Ver. 2.0 or later	---	---	Ver. 2.0 or later	---	---	---	659
	REGISTER COMPARISON TABLE	CTBL/ @CTBL	882	CTBL(882) is used to register a comparison table and perform comparisons for a high-speed counter PV.	---	Ver. 2.0 or later	---	---	OK	---	---	---	662
	SPEED OUTPUT	SPED/ @SPED	885	SPED(885) is used to set the output pulse frequency for a specific port and start pulse output without acceleration or deceleration.	---	Ver. 2.0 or later	---	---	OK	---	---	---	669
	SET PULSES	PULS/ @PULS	886	PULS(886) is used to set the pulse output amount (number of output pulses).	---	Ver. 2.0 or later	---	---	OK	---	---	---	673
	PULSE OUTPUT	PLS2/ @PLS2	887	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.	---	Ver. 2.0 or later	---	---	OK	---	---	---	675

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
High-speed Counter/ Pulse Output Instructions	ACCELERATION CONTROL	ACC/ @ACC	888	ACC(888) outputs pulses to the specified output port at the specified frequency using the specified acceleration and deceleration rate.	---	Ver. 2.0 or later	---	---	OK	---	---	---	684
	ORIGIN SEARCH	ORG/ @ORG	889	ORG(889) performs an origin search or origin return operation.	---	Ver. 2.0 or later	---	---	OK	---	---	---	692
	PWM outputs	PWM/ @PWM	891	PWM(891) is used to output pulses with a variable duty factor.	---	Ver. 2.0 or later	---	---	OK	---	---	---	695
	INTERRUPT FEEDIN	IFEED/@IFEED	892	IFEED(892) uses an input interrupt as a trigger to switch from speed control to position control and move the specified number of pulses. IFEED(892) is supported only by the CJ2M.	---	Ver. 2.0 or later	---	---	---	---	---	---	698
Step Instructions	STEP START	SNXT	009	SNXT(009) is used to control progression of step execution in the step program area.	OK	OK	OK	OK	OK	OK	OK	OK	702
	STEP DEFINE	STEP	008	STEP(008) is used to define the beginning and the end of the step program area.	OK	OK	OK	OK	OK	OK	OK	OK	702
Basic I/O Unit Instructions	I/O REFRESH	IORF/ @IORF	097	Refreshes the specified I/O words.	OK	OK	OK	OK	OK	OK	OK	OK	712
	SPECIAL I/O UNIT I/O REFRESH	FIORF/ @FIORF	225	Performs I/O refreshing immediately for the specified Special I/O Unit's allocated CIO Area and DM Area words with the specified unit number.	OK	OK	CJ1-H-R only	---	---	---	---	---	715
	CPU BUS UNIT I/O REFRESH	DLNK/ @DLNK	226	Performs I/O refreshing immediately for the CPU Bus Unit with the specified unit number.	OK	OK	OK	OK	OK	OK	OK	---	718
	7-SEGMENT DECODER	SDEC/ @SDEC	078	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.	OK	OK	OK	OK	OK	OK	OK	OK	722
	DIGITAL SWITCH INPUT	DSW	210	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.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	725
	TEN KEY INPUT	TKY	211	Reads numeric data from a ten-key keypad connected to an Input Unit and stores up to 8 digits of BCD data in the specified words.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	729
	HEXADECIMAL KEY INPUT	HYK	212	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.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	732
	MATRIX INPUT	MTR	213	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.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	736
	7-SEGMENT DISPLAY OUTPUT	7SEG	214	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.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	740
	ANALOG INPUT DIRECT CONVERSION	AIDC/@AIDC	216	Reads the input conversion value of the specified analog input number from the CJ1W-AD042 Analog Input Unit in Direct Conversion Mode.	Ver. 1.1 or later	OK	---	---	---	---	---	---	744
	ANALOG OUTPUT DIRECT CONVERSION	AODC/@AODC	217	Outputs the output set value for the specified analog output number to the CJ1W-DA042V Analog Output Unit in Direct Conversion Mode.	Ver. 1.1 or later	OK	---	---	---	---	---	---	747
	PCU HIGH-SPEED POSITIONING	NCDMV/@NCDMV	218	NCDMV(218) starts absolute or relative high-speed point-to-point positioning for the specified axis of a CJ1W-NC□□4, CJ1W-NC□81, or CJ1W-NC□82 Position Control Unit.	Ver. 1.3 or later	OK	---	---	---	---	---	---	750
	PCU POSITIONING TRIGGER	NCDTR/@NCDTR	219	NCDTR(219) is used to start a sequence for Memory Operation of a CJ1W-NC□81/NC□82 Position Control Unit when the start condition for the sequence is waiting for a command from NCDTR(219).	Ver. 1.3 or later	OK	---	---	---	---	---	---	754

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Basic I/O Unit Instructions	INTELLIGENT I/O READ	IORD/ @IORD	222	Reads the contents of memory area of a Special I/O Unit or CPU Bus Unit.	OK	OK	OK	OK	OK	OK	OK	OK	757
	INTELLIGENT I/O WRITE	IOWR/ @IOWR	223	Outputs the contents of the CPU Unit's I/O memory area to a Special I/O Unit or CPU Bus Unit.	OK	OK	OK	OK	OK	OK	OK	OK	760
Serial Communications Instructions	PROTOCOL MACRO	PMCR/ @PMCR	260	Starts a communications sequence (protocol data) that is registered in a Serial Communications Board (CS Series only) or Serial Communications Unit.	OK	OK	OK	OK	OK	OK	OK	OK	766
	TRANSMIT	TXD/ @TXD	236	Outputs the specified number of bytes of data from the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports. (The Serial Communications Board must be Ver. 1.2 or later).	OK	OK	OK	OK	OK	OK (See note 1.)	OK (See note 2.)	OK	773
	RECEIVE	RXD/ @RXD	235	Reads the specified number of bytes of data from the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports. (The Serial Communications Board must be Ver. 1.2 or later).	OK	OK	OK	OK	OK	OK (See note 1.)	OK (See note 2.)	OK	779
	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU/ @TXDU	256	Outputs the specified number of bytes of data from one of the Serial Communications Unit's serial ports. (The Serial Communications Unit must be Ver. 1.2 or later).	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 4.0 or later	---	---	789
	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU/ @RXDU	255	Reads the specified number of bytes of data from one of the Serial Communications Unit's serial ports. (The Serial Communications Unit must be Ver. 1.2 or later).	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 4.0 or later	---	---	796
	DIRECT TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	DTXDU/ @DTXDU	262	Outputs the specified number of bytes of data from the serial port of a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit. The data is output in no-protocol mode from the specified first word with the start code and end code (if any) specified in the allocated DM Setup Area. This instruction sends the data to the Serial Communications Unit as soon as the instruction is executed to achieve high-speed data transmission.	OK	OK	---	---	---	---	---	---	803
	DIRECT RECEIVE VIA SERIAL COMMUNICATIONS UNIT	DRXDU/ @DRXDU	261	Reads the specified number of bytes of data from the serial port of a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit to CPU Unit memory starting at the specified first word. The data is read in no-protocol mode with the start code and end code (if any) specified in the allocated DM Setup Area. This instruction reads the data from the Serial Communications Unit as soon as the instruction is executed to achieve high-speed data reception.	OK	OK	---	---	---	---	---	---	809
	CHANGE SERIAL PORT SETUP	STUP/ @STUP	237	Changes the communications parameters of a serial port on the CPU Unit, Serial Communications Board (CS Series only), or Serial Communications Unit (CPU Bus Unit).	OK	OK	OK	OK	OK	OK	OK	OK	817
	PROTOCOL MACRO 2	PMCR2/ @PMCR2	264	Starts a communications sequence (protocol data) that is registered in a Serial Communications Unit.	OK	OK	---	---	---	---	---	---	821

Note 1. Transmission and reception by specifying the serial communications port of the serial communications board are enabled in the CS1D-CPU□□SA with unit version 4.0 or later.

2. In the CS1D CPU Units (for Duplex-CPU Systems), transmission and reception by specifying the serial communications port of the serial communications board cannot be used.

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Network Instructions	NETWORK SEND	SEND/ @SEND	090	Sends data to a node in the network.	OK	OK	OK	OK	OK	OK	OK	OK	851
	NETWORK SEND 2	SEND2/ @SEND2	491	Sends data to a node on a network.	OK	OK	---	---	---	---	---	---	857
	NETWORK RECEIVE	RECV/ @RECV	098	Requests data to be transmitted from a node in the network and receives the data.	OK	OK	OK	OK	OK	OK	OK	OK	861
	NETWORK RECEIVE 2	RECV2/ @RECV2	492	Requests data to be transmitted from a node in the network and receives the data.	OK	OK	---	---	---	---	---	---	866
	DELIVER COMMAND	CMND/ @CMND	490	Sends an FINS command and receives the response.	OK	OK	OK	OK	OK	OK	OK	OK	870
	DELIVER COMMAND 2	CMND2/ @CMND2	493	Sends an FINS command and receives the response.	OK	OK	---	---	---	---	---	---	878
	EXPLICIT MESSAGE SEND	EXPLT	720	Sends an explicit message with any service code.	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	882
	EXPLICIT GET ATTRIBUTE	EGATR	721	Sends an information/status read command in an explicit message (Get Attribute Single, Service Code: 0E hex).	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	888
	EXPLICIT SET ATTRIBUTE	ESATR	722	Sends an information write command in an explicit message (Set Attribute Single, Service Code: 10 hex).	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	893
	EXPLICIT WORD READ	ECHRD	723	Reads data to the local CPU Unit from another CPU Unit in the network. (The remote CPU Unit must support explicit messages.)	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	897
	EXPLICIT WORD WRITE	ECHWR	724	Writes data from the local CPU Unit to another CPU Unit in the network. (The remote CPU Unit must support explicit messages.)	OK	OK	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	Ver. 2.0 or later	---	---	900
File Memory Instructions	READ DATA FILE	FREAD/ @FREAD	700	Reads the specified data or amount of data from the specified data file in file memory to the specified data area in the CPU Unit.	OK	OK	OK	OK	OK	OK	OK	OK	911
	WRITE DATA FILE	FWRIT/ @FWRIT	701	Overwrites or appends data in the specified data file in file memory with the specified data from the data area in the CPU Unit.	OK	OK	OK	OK	OK	OK	OK	OK	916
	WRITE TEXT FILE	TWRIT/ @TWRIT	704	Reads ASCII data from I/O memory and stores that data in the Memory Card as a text file (writing a new file or appending a file).	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	922
Display Instructions	Display Instructions	MSG/ @MSG	046	Reads the specified sixteen words of extended ASCII and displays the message on a Peripheral Device such as a Programming Console.	OK	OK	OK	OK	OK	OK	OK	OK	927
Clock Instructions	CALENDAR ADD	CADD/ @CADD	730	Adds time to the calendar data in the specified words.	OK	OK	OK	OK	OK	OK	OK	OK	930
	CALENDAR SUBTRACT	CSUB/ @CSUB	731	Subtracts time from the calendar data in the specified words.	OK	OK	OK	OK	OK	OK	OK	OK	930
	HOURS TO SECONDS	SEC/ @SEC	065	Converts time data in hours/minutes/seconds format to an equivalent time in seconds only.	OK	OK	OK	OK	OK	OK	OK	OK	935
	SECONDS TO HOURS	HMS/ @HMS	066	Converts seconds data to an equivalent time in hours/minutes/seconds format.	OK	OK	OK	OK	OK	OK	OK	OK	937
	CLOCK ADJUSTMENT	DATE/ @DATE	735	Changes the internal clock setting to the setting in the specified source words.	OK	OK	OK	OK	OK	OK	OK	OK	939
Debugging Instructions	Trace Memory Sampling	TRSM	045	When TRSM(045) is executed, the status of a preselected bit or word is sampled and stored in Trace Memory.	OK	OK	OK	OK	OK	OK	OK	OK	941
Failure Diagnosis Instructions	FAILURE ALARM	FAL/ @FAL	006	Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop PLC operation.	OK	OK	OK	OK	OK	OK	OK	---	944
	SEVERE FAILURE ALARM	FALS	007	Generates user-defined fatal errors. Fatal errors stop PLC operation.	OK	OK	OK	OK	OK	OK	OK	---	951
	FAILURE POINT DETECTION	FPD	269	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.	OK	OK	OK	OK	OK	OK	OK	OK	957

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Other Instructions	SET CARRY	STC/ @STC	040	Sets the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	965
	CLEAR CARRY	CLC/ @CLC	041	Turns OFF the Carry Flag (CY).	OK	OK	OK	OK	OK	OK	OK	OK	965
	SELECT EM BANK	EMBC/ @EMBC	281	Changes the current EM bank.	OK	OK	OK	OK	OK	OK	OK	OK	966
	EXTEND MAXIMUM CYCLE TIME	WDT/ @WDT	094	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.	OK	OK	OK	OK	OK	OK	OK	OK	968
	SAVE CONDITION FLAGS	CCS/ @CCS	282	Saves the current status of the Condition Flags in a separate area within the CPU Unit.	OK	OK	OK	OK	OK	OK	OK	---	970
	LOAD CONDITION FLAGS	CCL/ @CCL	283	Restores the status of the Condition Flags that were saved in a separate area within the CPU Unit by CCS(282).	OK	OK	OK	OK	OK	OK	OK	---	970
	CONVERT ADDRESS FROM CV	FRMCV/ @FRMCV	284	Converts a CV-series PLC memory address to its corresponding CS/CJ-series PLC memory address.	OK	OK	OK	OK	OK	OK	OK	---	973
	CONVERT ADDRESS TO CV	TOCV/ @TOCV	285	Converts a CS/CJ-series PLC memory address to its corresponding CV-series PLC memory address.	OK	OK	OK	OK	OK	OK	OK	---	977
	DISABLE PERIPHERAL SERVICING	IOSP/ @IOSP	287	Disables peripheral servicing during program execution in Parallel Processing Mode or Peripheral Servicing Priority Mode.	OK	OK	OK	OK	OK	OK	---	---	980
	ENABLE PERIPHERAL SERVICING	IORS	288	Enables the peripheral servicing during program execution in Parallel Processing Mode that was disabled by IOSP(287), the DISABLE PERIPHERAL SERVICING instruction.	OK	OK	OK	OK	OK	OK	---	---	980
Block Programming Instructions	BLOCK PROGRAM BEGIN	BPRG	096	Indicates the beginning of the block program area specified by number.	OK	OK	OK	OK	OK	OK	OK	OK	986
	BLOCK PROGRAM END	BEND	801	Indicates the end of the block program area.	OK	OK	OK	OK	OK	OK	OK	OK	986
	BLOCK PROGRAM PAUSE	BPPS	811	Pause the specified block program from another block program.	OK	OK	OK	OK	OK	OK	OK	OK	988
	BLOCK PROGRAM RESTART	BPRS	812	Restart the specified block program from another block program.	OK	OK	OK	OK	OK	OK	OK	OK	988
	CONDITIONAL BLOCK EXIT	EXIT	806	When the execution condition is ON (when an operand is not specified), or when the specified bit is ON (when an operand is specified), the instructions from the EXIT instruction to the BEND instruction are not executed and the block program is exited.	OK	OK	OK	OK	OK	OK	OK	OK	990
	CONDITIONAL BLOCK EXIT NOT	EXIT NOT	806	When the specified bit is OFF, the instructions from the EXIT NOT instruction to the BEND instruction are not executed and the block program is exited.	OK	OK	OK	OK	OK	OK	OK	OK	990
	Branching	IF	802	When the execution condition (when an operand is not specified) or the specified bit (when an operand is specified) is OFF, the next and following instructions are executed. When ON, instructions up to ELSE are disregarded.	OK	OK	OK	OK	OK	OK	OK	OK	992
		IF NOT	802	When the specified bit is ON, the next and following instructions are executed. When OFF, instructions up to ELSE are disregarded.	OK	OK	OK	OK	OK	OK	OK	OK	992
		ELSE	803	Indicates the beginning of the block that is executed when IF is false.	OK	OK	OK	OK	OK	OK	OK	OK	992
		IEND	804	Indicates the end of the conditional branch block area.	OK	OK	OK	OK	OK	OK	OK	OK	992
	ONE CYCLE AND WAIT	WAIT	805	The instructions from the WAIT instruction to the BEND instruction are not executed until the execution condition (when an operand is not specified) or the specified bit (when an operand is specified) turns ON.	OK	OK	OK	OK	OK	OK	OK	OK	995
	ONE CYCLE AND WAIT NOT	WAIT NOT	805	Stops execution of the rest of the block program until a specified bit OFF or an operand bit turns OFF	OK	OK	OK	OK	OK	OK	OK	OK	995

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Block Programming Instructions	HUNDRED-MS TIMER WAIT	TIMW	813	Delays execution of the rest of the block program until the specified time has elapsed.	OK	OK	OK	OK	OK	OK	OK	OK	998
		TIMWX	816		OK	OK	OK	OK	OK	OK	OK	---	998
	COUNTER WAIT	CNTW	814	Delays execution of the rest of the block program until the specified count has been achieved.	OK	OK	OK	OK	OK	OK	OK	OK	1001
		CNTWX	818		OK	OK	OK	OK	OK	OK	OK	---	1001
	TEN-MS TIMER WAIT	TMHW	815	Delays execution of the rest of the block program until the specified time has elapsed.	OK	OK	OK	OK	OK	OK	OK	OK	1003
		TMHWX	817		OK	OK	OK	OK	OK	OK	OK	---	1003
	Loop Control	LOOP	809	Indicates the beginning of the loop (LOOP to LEND).	OK	OK	OK	OK	OK	OK	OK	OK	1006
		LEND	810	Indicates the end of the loop (LOOP to LEND). LOOP to LEND is repeated until the execution condition (when an operand is not specified) or the specified bit (when an operand is specified) turns ON.	OK	OK	OK	OK	OK	OK	OK	OK	1006
		LEND NOT	810	Indicates the end of the loop (LOOP to LEND NOT). LOOP to LEND NOT is repeated until the specified bit turns OFF.	OK	OK	OK	OK	OK	OK	OK	OK	1006
Text String Processing Instructions	MOV STRING	MOV$/ @MOV$	664	Transfers a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1011
	CONCATENATE STRING	+$/ @+$	656	Links one text string to another text string.	OK	OK	OK	OK	OK	OK	OK	OK	1012
	GET STRING LEFT	LEFT$/ @LEFT$	652	Fetches a designated number of characters from the left (beginning) of a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1014
	GET STRING RIGHT	RGHT$/ @RGHT$	653	Reads a designated number of characters from the right (end) of a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1014
	GET STRING MIDDLE	MID$/ @MID$	654	Reads a designated number of characters from any position in the middle of a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1017
	FIND IN STRING	FIND$/ @FIND$	660	Finds a designated text string from within a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1019
	STRING LENGTH	LEN$/ @LEN$	650	Calculates the length of a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1021
	REPLACE IN STRING	RPLC$/ @RPLC$	661	Replaces a text string with a designated text string from a designated position.	OK	OK	OK	OK	OK	OK	OK	OK	1023
	DELETE STRING	DEL$/ @DEL$	658	Deletes a designated text string from the middle of a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1025
	EXCHANGE STRING	XCHG$/ @XCHG$	665	Replaces a designated text string with another designated text string.	OK	OK	OK	OK	OK	OK	OK	OK	1027
	CLEAR STRING	CLR$/ @CLR$	666	Clears an entire text string with NUL (00 hex).	OK	OK	OK	OK	OK	OK	OK	OK	1029
	INSERT INTO STRING	INS$/ @INS$	657	Inserts a designated text string into the middle of a text string.	OK	OK	OK	OK	OK	OK	OK	OK	1031
	String Comparison	LD, AND, OR+=$	670	String 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.	OK	OK	OK	OK	OK	OK	OK	OK	1033
		LD, AND, OR+<>$	671		OK	OK	OK	OK	OK	OK	OK	OK	1033
		LD, AND, OR+<$	672		OK	OK	OK	OK	OK	OK	OK	OK	1033
		LD, AND, OR+>$	674		OK	OK	OK	OK	OK	OK	OK	OK	1033
		LD, AND, OR+>=$	675		OK	OK	OK	OK	OK	OK	OK	OK	1033
Task Control Instructions	TASK ON	TKON/ @TKON	820	Makes the specified task executable.	OK	OK	OK	OK	OK	OK	OK	OK	1038
	TASK OFF	TKOF/ @TKOF	821	Puts the specified task into standby status.	OK	OK	OK	OK	OK	OK	OK	OK	1038
Model Conversion Instructions	BLOCK TRANSFER	XFERC/ @XFERC	565	Transfers the specified number of consecutive words.	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	---	---	---	1044
	SINGLE WORD DISTRIBUTE	DISTC/ @DISTC	566	Transfers the source word to a destination word calculated by adding an offset value to the base address.	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	---	---	---	1046
	DATA COLLECT	COLLC/ @COLLC	567	Transfers the source word (calculated by adding an offset value to the base address) to the destination word.	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	---	---	---	1049
	MOVE BIT	MOVBC/ @MOVBC	568	Transfers the specified bit.	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	---	---	---	1052
	BIT COUNTER	BCNTC/ @BCNTC	621	Counts the total number of ON bits in the specified word(s).	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	---	---	---	1054

Classification	Instruction	Mnemonic	Fun code	Function	CJ2H	CJ2M	CJ1-H/ CS1-H	CJ1M		CS1D		CJ1/ CS1	Page
								-CPU1□	-CPU2□	-SA/S	-HA/H		
Special Function Block Instructions	GET VARIABLE ID	GETID/ @GETID	286	Outputs the FINS command variable type (data area) code and word address for the specified variable or address.	OK	OK	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 3.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1056
SFC Instructions	STEP ACTIVATION	SA/ @SA	784	Makes the specified step or subchart active to start execution of the actions.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1060
	STEP DEACTIVATION	SE/ @SE	785	Makes the specified step or subchart inactive to end execution of the actions.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1060
	READ SET TIMER	TSR/ @TSR	780	The present value of the Step Timer specified by S is stored starting at D.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1062
	SET STEP TIMER	TSW/ @TSW	781	The present value of the Step Timer specified by S is changed to the value specified starting at D.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1062
	SFC ON	SFCON	789	Restarts execution of an SFC task that was ended or paused using one of the other SFC Task Control Instructions.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1064
	SFC OFF	SFCOFF	790	Ends execution of an SFC task. The status of all outputs is held. When execution of the SFC task is restarted, it is executed from the initial step.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1064
	SFC PAUSE WITH RESET	SFCPR	793	Pauses execution of an SFC task. The status of all outputs is reset. When execution of a paused task is restarted, execution will start from the step that was active when the task was paused.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1066
	SFC PAUSE WITH NO RESET	SFCPRN	791	Pauses execution of an SFC task. The status of all outputs is held. When execution of a paused task is restarted, execution will start from the step that was active when the task was paused.	OK	OK	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	Ver. 4.0 or later	---	1066

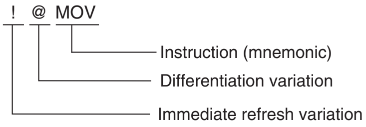
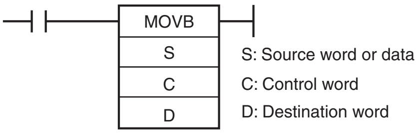
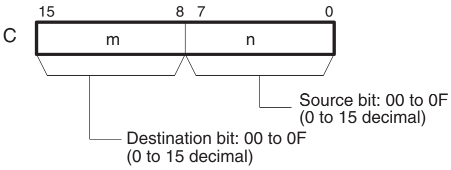
SECTION 3
Instructions

This section describes each of the instructions that can be used in programming NSJ/CS/CJ-series PLCs. Instructions are described in order of function, as classified in *Section 2 Summary of Instructions.*

Notation and Layout of Instruction Descriptions

Instructions are described in groups by function. Refer to Appendix A List of Instructions by Function Code for a list of instructions by mnemonic that lists the page number in this section for each instruction.

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

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Instruction</td><td>Indicates the name of the instruction. Example: MOVE BIT</td></tr>
<tr><td>Mnemonic</td><td>Indicates the mnemonic.
Example: MOVB(082)</td></tr>
<tr><td>Variations</td><td>Differentiation
@ Instruction that differentiates when the execution condition turns ON.
% 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.
 </td></tr>
<tr><td>Function code</td><td>Indicates the function code.</td></tr>
<tr><td>Function</td><td>The basic purpose of the instruction is described after the section heading.</td></tr>
<tr><td>Symbol</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.
 </td></tr>
<tr><td>Applicable Program Areas</td><td>The program areas in which the instruction can be used are specified. “OK” indicates the areas in which the instruction can be used.
<table>
<tr><th>Area</th><th>Function block definitions</th><th>Block program areas</th><th>Step program areas</th><th>Subroutines</th><th>Interrupt tasks</th><th>SFC action or transition programs</th></tr>
<tr><th>Usage</th><td>OK</td><td>OK</td><td>OK</td><td>OK</td><td>OK</td><td>OK</td></tr>
</table></td></tr>
<tr><td>Operands</td><td>Indicates a description of the operand, the data type, and the size.
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 above 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 rowspan="2">Area</th><th colspan="8">Word addresses</th><th colspan="2">Indirect DM/EM addresses</th><th rowspan="2">Con-stants</th><th colspan="3">Registers</th><th colspan="2">Flags</th><th rowspan="2">Pulse bits</th><th rowspan="2">TR bits</th></tr>
<tr><th>CIO</th><th>WR</th><th>HR</th><th>AR</th><th>T</th><th>C</th><th>DM</th><th>EM</th><th>@DM
@EM</th><th>*DM
*EM</th><th>DR</th><th>IR</th><th>Indirect using IR</th><th>TK</th><th>CF</th></tr>
<tr><td>S</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="3">OK</td><td rowspan="2">OK</td><td rowspan="3">OK</td><td rowspan="3">---</td><td rowspan="3">OK</td><td rowspan="3">---</td><td rowspan="3">---</td><td rowspan="3">---</td><td rowspan="3">---</td></tr>
<tr><td>C</td></tr>
<tr><td>D</td><td>---</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></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>OFF</td></tr>
<tr><td>Equal Flag</td><td>=</td><td>• ON if the data being transferred (D) is 0.
• OFF in all other cases.</td></tr>
<tr><td>Negative Flag</td><td>N</td><td>• ON if the leftmost bit of the data being transferred (D) is 1.
• OFF in all other cases.</td></tr>
</table></td></tr>
<tr><td>Function</td><td>Indicates the function of the instruction.</td></tr>
<tr><td>Hint</td><td>Indicates a supplemental explanation of other than the main function.</td></tr>
<tr><td>Precautions</td><td>Indicates important points when using an instruction.</td></tr>
<tr><td>Example Programming</td><td>One or more examples of using the instruction with specific operands is provided to further explain the function of the instruction.</td></tr>
</table>

Constants

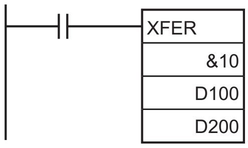
Constants input for operands are given as listed below.

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 Programming Devices are given in the following table.

Operand	CX-Programmer	Programming Console
Operands specifying bit strings (normally input as hexadecimal)	Input as decimal with an & prefix or input as hexadecimal with an # prefix. (See note.)	The Cont/# Key can be pressed to input hexadecimal values by default with an # prefix.
Operands specifying numeric values (normally input as decimal)		The CHG Key can then be pressed to rotate between hexadecimal (with # prefix), signed decimal (with +/–), and unsigned decimal (with & prefix).
Operands specifying control numbers (except for jump numbers)	Input as decimal with an # prefix. (See note.)	Input directly in decimal form. • If the & prefix is automatically added, the CHG Key can be pressed to rotate between unsigned decimal (with & prefix), hexadecimal (with # prefix), and signed decimal (with +/–). • If no prefix is displayed, the value must be entered in decimal form.

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

Condition Flags

Programming Console labels are used for condition flags in this section. With the CX-Programmer, the condition flags are registered in advance as global symbols with “P_” in front of the symbol name.

Flag	CX-Programmer label	Programming Console label
Error Flag	P_ER	ER
Access Error Flag	P_AER	AER
Carry Flag	P_CY	CY
Greater Than Flag	P_GT	>
Equals Flag	P_EQ	=
Less Than Flag	P_LT	<
Negative Flag	P_N	N
Overflow Flag	P_OF	OF
Underflow Flag	P_UF	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	ON
Always OFF Flag	P_Off	OFF

Symbol Instructions

Some of the C/CV-series PLC instructions have been changed to different instructions with the same functionality for the CS/CJ-series PLCs.

Instruction group	C/CV Series	CS/CJ Series
Sequence Control	JMP #0 / JME #0	JMP0 / JME0
Comparison	EQU	AND=
Data Movement	MOVQ	MOV
Increment/Decrement	INC	++B
	INCL	++BL
	INCB	++
	INBL	++L
	DEC	--B
	DECL	--BL
	DECB	--
	DCBL	--L
Symbol Math	ADB	+C
	ADBL	+CL
	ADD	+BC
	ADDL	+BCL
	SBB	-C
	SBBL	-CL
	SUB	-BC
	SUBL	-BCL
	MBS	*
	MBSL	*L
	MLB	*U
	MUL	*B
	MULL	*BL
	DBS	/
	DBSL	/L
	DVB	/U
	DIV	/B
	DIVL	/BL
Interrupt Control	INT	MSKS / MSKR / CLIDI / EI

Sequence Input Instructions

Differentiated and Immediate Refreshing Instructions

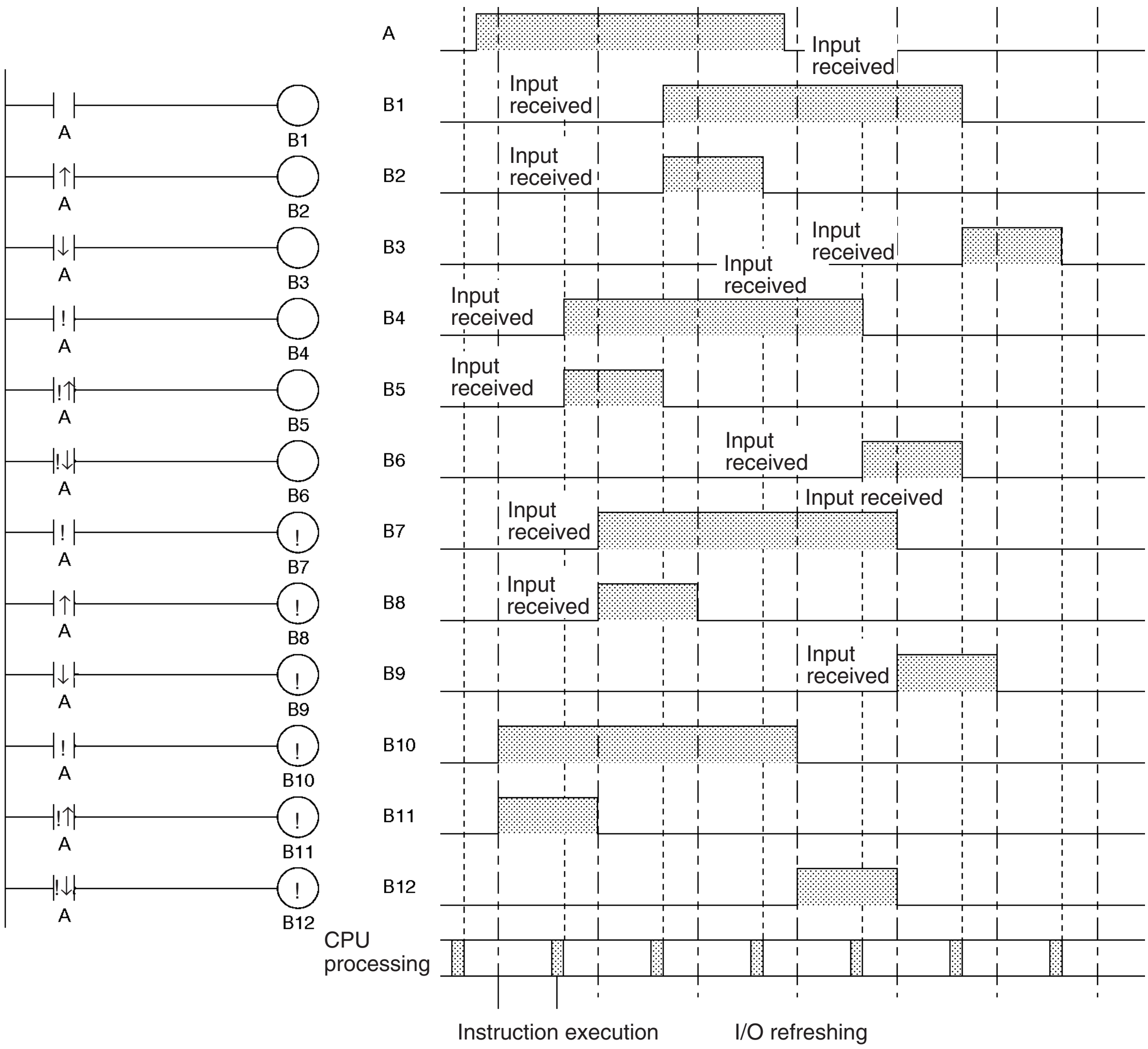
- The LOAD, AND, and OR instructions have differentiated and immediate refreshing variations in addition to their ordinary forms, and there are also two combinations available.
- The LOAD NOT, AND NOT, OR NOT, OUT, and OUT NOT instructions have immediate refreshing variations in addition to their ordinary 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 Unit's internal memory and the I/O Unit.
- 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. When using immediate-refresh versions of bit instructions, all 16 bits in the word containing the specified bit will be refreshed.

  Immediate refresh instructions cannot be used for Units on Slave Racks.

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.	

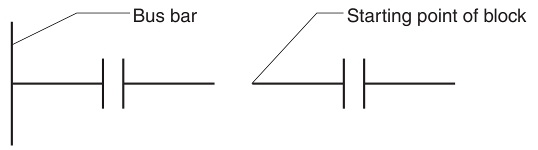
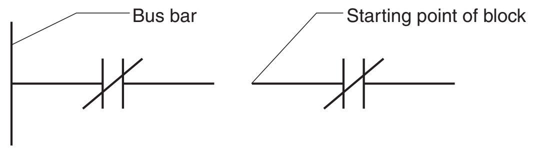
Operation Timing for I/O Instructions

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



LD/LD NOT

Instruction	Mnemonic	Variations	Function code	Function
LOAD	LD	@LD, %LD, !LD, !@LD, !%LD	---	Indicates a logical start and creates an ON/OFF execution condition based on the ON/OFF status of the specified operand bit.
LOAD NOT	LD NOT	@LD NOT, %LD NOT, ! LD NOT, !@LD NOT, !%LD NOT	---	Indicates a logical start and creates an ON/OFF execution condition based on the reverse of the ON/OFF status of the specified operand bit.

	LD	LD NOT
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
---	---	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
LD bit operand	OK	OK	OK	OK	OK	OK	OK[*1]	OK[*1]	---	---	---	---	---	OK	OK	OK	OK	OK
LD NOT bit operand																		---

[*1] CJ2 CPU Units only.

Flags

There are no flags affected by this instruction.

Function

● LD

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 NOT

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.

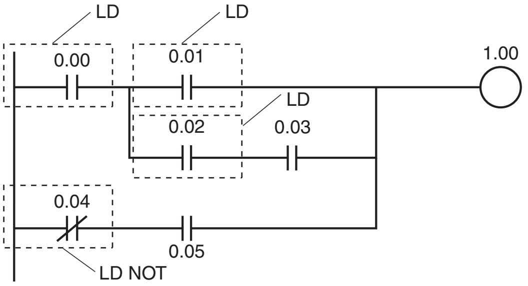
Hint

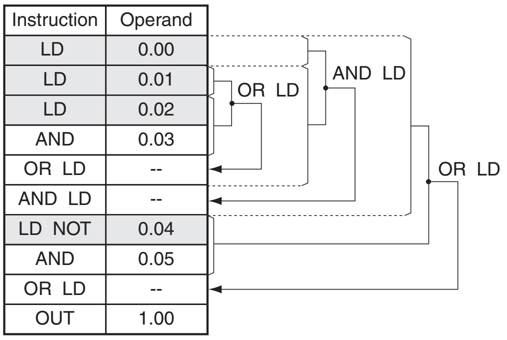
- LD/LD NOT is used in the following circumstances as an instruction for indicating a logical start.
  1. When directly connecting to the bus bar.
  2. When logic blocks are connected by AND LD or OR LD, i.e., at the beginning of a logic block. The AND LOAD and OR LOAD instructions are used to connect 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 Peripheral Device.
- When logic blocks are connected by AND LOAD or OR LOAD instructions, the total number of AND LOAD/OR LOAD instructions must match the total number of LOAD/LOAD NOT instructions minus1. If they do not match, a programming error will occur. For details, refer to 3-3-7 AND LOAD: AND LD and 3-3-8 OR LOAD: OR LD.

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/LD NOT. An immediate refresh instruction updates the status of the input bit just before the instruction is executed for Basic Input Units (but not Basic Input Units on Slave Racks or for C200H Group 2 Multi-point Input Units).
- For LD, 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 Basic Input 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.
- With CJ2 CPU Units, LD and LD NOT can be used for bits in the DM or EM Area. For other models of CPU Unit, LD and LD NOT cannot be used for bits in the DM or EM Area. Use LD TST(350) and LD TSTN(351) instead.

Example Programming





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	1.00

AND/AND NOT

Instruction	Mnemonic	Variations	Function code	Function
AND	AND	@AND, %AND, !AND, !@AND, !%AND	---	Takes a logical AND of the status of the specified operand bit and the current execution condition.
AND NOT	AND NOT	@AND NOT, %AND NOT, !AND NOT, !@AND NOT, !%AND NOT	---	Reverses the status of the specified operand bit and takes a logical AND with the current execution condition.

Symbol	AND	AND NOT
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
---	---	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
AND bit operand	OK	OK	OK	OK	OK	OK	OK*1	OK*1	---	---	---	---	---	OK	OK	OK	OK	---
AND NOT bit operand																		

*1 CJ2 CPU Units only.

Flags

There are no flags affected by this instruction.

Function

● AND

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 Basic Input Unit's input terminal is read.

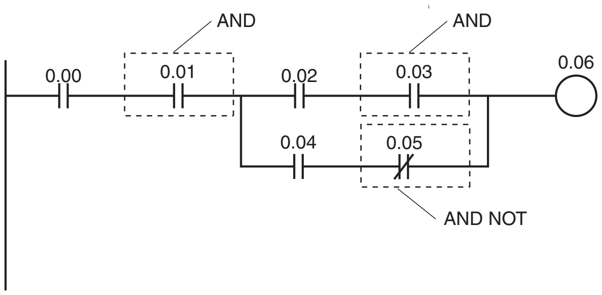
● AND NOT

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 Basic Input Unit's input terminals is read.

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/AND NOT. An immediate refresh instruction updates the status of the input bit just before the instruction is executed from the Basic Input Unit (but not Basic Input Units on Slave Racks or for C200H Group 2 Multi-point Input Units).
- 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 Basic Input 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.
  With CJ2 CPU Units, AND and AND NOT can be used for bits in the DM or EM Area. For other models of CPU Unit, AND and AND NOT cannot be used for bits in the DM or EM Area. Use AND TST (350) instead.

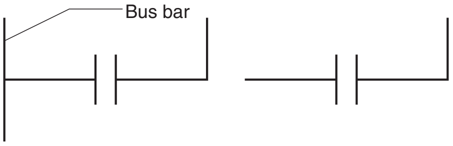
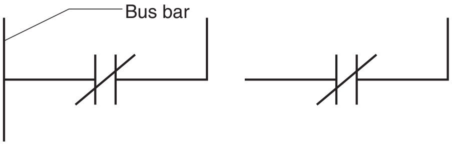
Example Programming



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	0.06

OR/OR NOT

Instruction	Mnemonic	Variations	Function code	Function
OR	OR	@OR, %OR, !OR, !@OR, !%OR	---	Takes a logical OR of the ON/OFF status of the specified operand bit and the current execution condition.
OR NOT	OR NOT	@OR NOT, %OR NOT, !OR NOT, !@OR NOT, !%OR NOT	---	Reverses the status of the specified bit and takes a logical OR with the current execution condition.

	OR	OR NOT
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
---	---	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
OR bit operand OR NOT bit operand	OK	OK	OK	OK	OK	OK	OK[*1]	OK[*1]	---	---	---	---	---	OK	OK	OK	OK	---

*1 CJ2 CPU Units only.

Flags

There are no flags affected by this instruction.

Function

● OR

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 Basic Input Unit's input terminal is read.

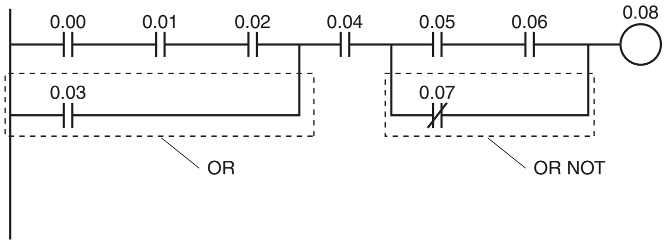
● OR NOT

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 Basic Input Unit's input terminal is read.

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/OR NOT. An immediate refresh instruction updates the status of the input bit just before the instruction is executed from the Basic Input Unit (but not for Basic Input Units on Slave Racks or for C200H Group 2 Multi-point Input Units).
- 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 Basic Input 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.
- With CJ2 CPU Units, OR and OR NOT can be used for bits in the DM or EM Area. For other models of CPU Unit, OR and OR NOT cannot be used for bits in the DM or EM Area. Use OR TST (350) instead.

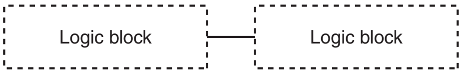
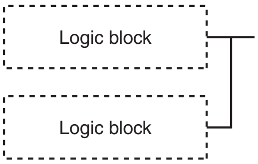
Example Programming



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	0.08

AND LD/OR LD

Instruction	Mnemonic	Variations	Function code	Function
AND LOAD	AND LD	---	---	Takes a logical AND between logic blocks.
OR LOAD	OR LD	---	---	Takes a logical OR between logic blocks.

	AND LD	OR LD
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

There are no flags affected by this instruction.

Function

● AND LD

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

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

LD
to
Logic block A

LD
to
Logic block B

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

● OR LD

OR 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.

LD
to
Logic block A

LD
to
Logic block B

OR LD ······ Parallel connection between logic block A and logic block B.

Hint

● AND LD

- 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.

● OR LD

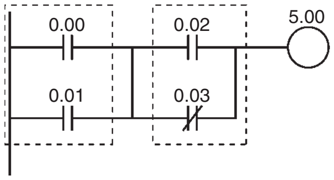
- Three or more logic blocks can be connected in parallel using this instruction to first connect two of the logic blocks and then to connect the next and subsequent ones in order. It is also possible to continue placing this instruction after three or more logic blocks and connect them together in parallel.

Precautions

- 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.
- The CPU Unit performs logic operations between logic blocks. The AND LD and OR LD instructions perform logic operations between the logic blocks based on those results to output the result of the entire rung.

● AND LD

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).



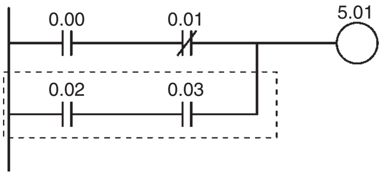
Coding

Instruction	Operand
LD	0.00
OR	0.01
LD	0.02
OR NOT	0.03
AND LD	---
OUT	5.00

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

● OR LD

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.



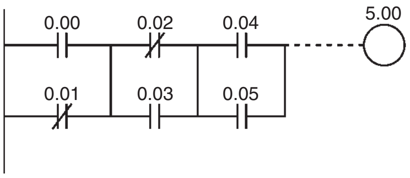
Coding

Instruction	Operand
LD	0.00
AND NOT	0.01
LD	0.02
AND	0.03
OR LD	---
OUT	5.01

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

Example Programming

● AND LD



Coding Example (1)

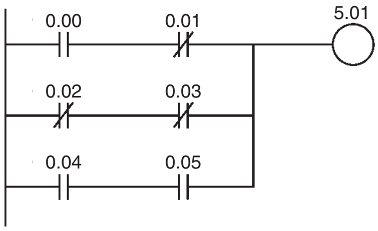
Instruction	Operand
LD	0.00
OR NOT	0.01
LD NOT	0.02
OR	0.03
AND LD	---
LD	0.04
OR	0.05
AND LD	---
. .	. .
OUT	5.00

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
. .	. .
AND LD	---
AND LD	---
. .	. .
OUT	5.00

- 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 Peripheral Device.

● OR LD



Coding Example (1)

Instruction	Operand
LD	0.00
AND NOT	0.01
LD NOT	0.02
AND NOT	0.03
OR LD	---
LD	0.04
AND	0.05
OR LD	---
. .	. .
OUT	5.01

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	5.01

- The OR LOAD instruction can be used repeatedly. In programming method (2) above, however, the number of OR LOAD instructions becomes one less than the number of LOAD and LOAD NOT instructions before that.
- In method (2), make sure that the total number of LOAD and LOAD NOT instructions before OR LOAD is not more than eight.
- To use nine or more, program using method (1).
- If there are nine or more with method (2), then a program error will occur during the program check by the Peripheral Device.

NOT

Instruction	Mnemonic	Variations	Function code	Function
NOT	NOT	---	520	Reverses the execution condition.

Symbol	NOT
	NOT(520)

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

There are no flags affected by NOT(520).

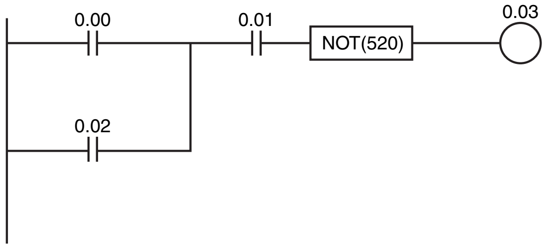
Function

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

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).
- The CPU Unit performs logic operations between logic blocks. When you use the NOT instruction and there are multiple input instructions between the left bus bar and the NOT instruction, create the rung so that the signal flow (i.e., the power flow) goes from left to right on the ladder diagram.

Example Programming



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

0.00	0.01	0.02	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

UP/DOWN

Instruction	Mnemonic	Variations	Function code	Function
CONDITION ON	UP	---	521	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.
CONDITION OFF	DOWN	---	522	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.

Symbol	UP	DOWN
	UP(521)	DOWN(522)

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

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

Function

● UP

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

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.

Hint

- 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.

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.

Note Observe the following precaution when using UP(521) in a function block definition.

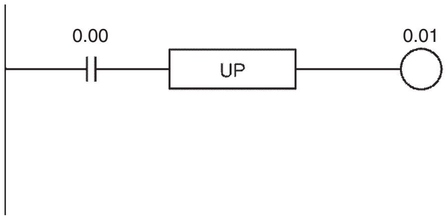
- The operation of UP(521) and DOWN(522) will not be consistent if the same function block instance is executed more than once in the same cycle.
- An instance will not be executed while EN is OFF. Caution is thus required when using UP(521) and DOWN(522) in a function block definition. For details, refer to information on restrictions on using ladder programming instructions in the CX-Programmer Operation Manual: Function Blocks.

Note Observe the following precaution when using UP(521) and DOWN(522) in a subroutine.

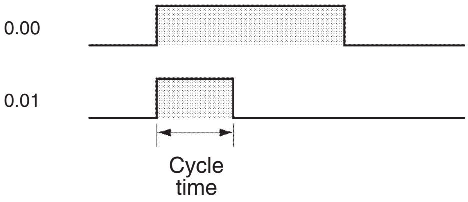
- The operation of UP(521) and DOWN(522) will not be consistent if the same subroutine is executed more than once in the same cycle.
- An subroutine will not be executed while the input condition for the subroutine is OFF. Caution is thus required when using UP(521) and DOWN(522) in a function block definition. For details, refer to information on SBS(091).
- The CPU Unit performs logic operations between logic blocks. When you use the UP or DOWN instruction and there are multiple input instructions between the left bus bar and the UP or DOWN instruction, create the rung so that the signal flow (i.e., the power flow) goes from left to right on the ladder diagram.
- The UP or DOWN instruction takes the previous execution result as the input signal and detects when the signal turns ON or turns OFF (upward or downward differentiation). If either of these instructions is used for a logic block, upward or downward differentiation is performed for the range of the logic block.

Example Programming

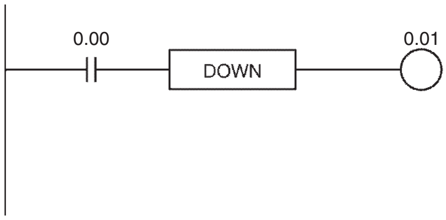
● UP



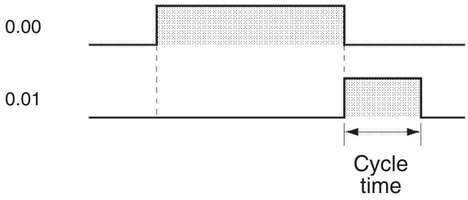
When CIO 0.00 goes from OFF to ON, CIO 0.01 is turned ON for just one cycle.



● DOWN

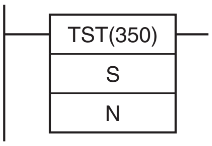
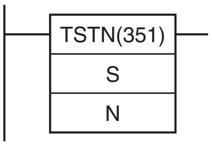


When CIO 0.00 goes from ON to OFF, CIO 0.01 is turned ON for just one cycle.



LD TST/LD TSTN

Instruction	Mnemonic	Variations	Function code	Function
LOAD BIT TEST	LD TST	---	350	LD TST(350) is used in the program like LD; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.
LOAD BIT TEST NOT	LD TSTN	---	351	LD TSTN(351) is used in the program like LD NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.

	LD TST	LD TSTN
Symbol	 S: Source word N: Bit number	 S: Source word N: Bit number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
N	Bit number	UINT	1

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
N											OK							

Flags

Name	Label	Operation
Error Flag	ER	OFF or unchanged [*1]
Equals Flag	=	OFF or unchanged [*1]
Negative Flag	N	OFF or unchanged [*1]

*1 In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.
In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, these Flags are left unchanged.

Function

● LD TST

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

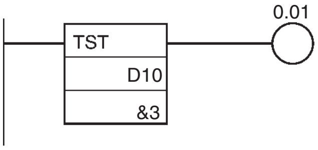
● LD TSTN

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

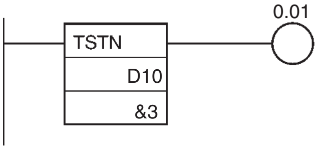
Precautions

- 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).

Example Programming



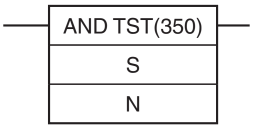
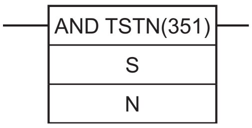
In the left example, CIO 0.01 is turned ON when bit 3 of D10 is ON.



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

AND TST/AND TSTN

Instruction	Mnemonic	Variations	Function code	Function
AND BIT TEST	AND TST	---	350	AND TST(350) is used in the program like AND; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.
AND BIT TEST NOT	AND TSTN	---	351	AND TSTN(351) is used in the program like AND NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.

	AND TST	AND TSTN
Symbol	 S: Source word N: Bit number	 S: Source word N: Bit number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
N	Bit number	UINT	1

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
N											OK							

Flags

Name	Label	Operation
Error Flag	ER	OFF or unchanged *1
Equals Flag	=	OFF or unchanged *1
Negative Flag	N	OFF or unchanged *1

*1 In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.
In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, these Flags are left unchanged.

Function

● AND TST

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

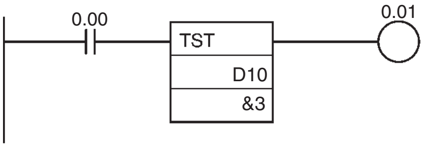
● AND TSTN

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

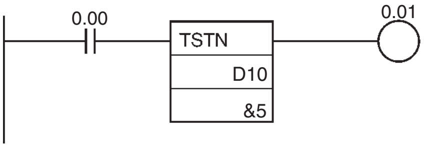
Precautions

- 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).

Example Programming



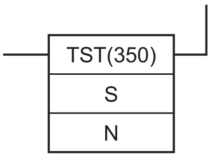
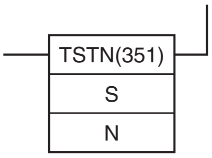
In the left example, CIO 0.01 is turned ON when CIO 0.00 and bit 3 of D10 are both ON.



In the left example, CIO 0.01 is turned ON when CIO 0.00 is ON and bit 5 of D10 is OFF.

OR TST/OR TSTN

Instruction	Mnemonic	Variations	Function code	Function
OR BIT TEST	OR TST	---	350	OR TST(350) is used in the program like OR; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.
OR BIT TEST NOT	OR TSTN	---	351	OR TSTN(351) is used in the program like OR NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.

	OR TST	OR TSTN
Symbol	 S: Source word N: Bit number	 S: Source word N: Bit number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
N	Bit number	UINT	1

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
N											OK							

Flags

Name	Label	Operation
Error Flag	ER	OFF or unchanged [*1]
Equals Flag	=	OFF or unchanged [*1]
Negative Flag	N	OFF or unchanged [*1]

[*1] In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.
In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, these Flags are left unchanged.

Function

● OR TST

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

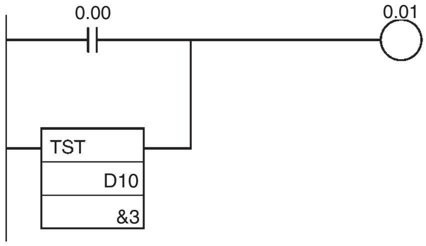
● OR TSTN

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

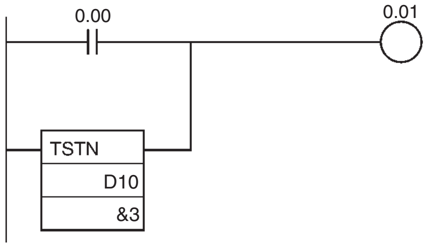
Precautions

- 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).

Example Programming



In the left example, CIO 0.01 is turned ON when CIO 0.00 or bit 3 of D10 is ON.

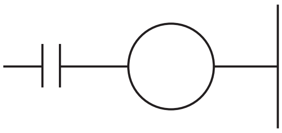
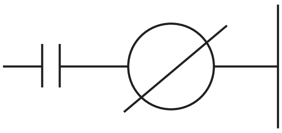


In the left example, CIO 0.01 is turned ON when CIO 0.00 is ON or bit 3 of D10 is OFF.

Sequence Output Instructions

OUT/OUT NOT

Instruction	Mnemonic	Variations	Function code	Function
OUTPUT	OUT	!OUT	---	Outputs the result (execution condition) of the logical processing to the specified bit.
OUTPUT NOT	OUT NOT	!OUT NOT	---	Reverses the result (execution condition) of the logical processing, and outputs it to the specified bit.

	OUT	OUT NOT
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
---	---	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
OUT OUT NOT	OK	OK	OK	OK	---	---	OK*1	OK*1	---	---	---	---	---	OK	---	---	---	OK

*1 CJ2 CPU Units only.

Flags

There are no flags affected by this instruction.

Function

● OUT

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 Basic Output Unit's output terminal in addition to the output bit in I/O memory.

● OUT NOT

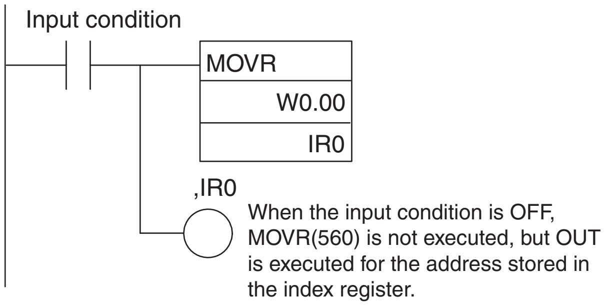
If there is no immediate refreshing specification, the status of the execution condition (power flow) is reversed and written to a specified bit in I/O memory. If there is an immediate refreshing specification, the status of the execution condition (power flow) is reversed and also written to the Basic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

Hint

- Immediate refreshing (!) can be specified for OUT and OUT NOT. An immediate refresh instruction updates the status of the output terminal just after the instruction is executed for the Basic Output Unit (but not for Basic Output Units on Slave Racks or for C200H Group 2 Multi-point Input Units), at the same time as it writes the status of the execution condition (power flow) to the specified output bit in I/O memory.
- Difference between SET/RSET and OUT
  For OUT, the operand bit is turned ON when the input condition turns ON and is turned OFF when the input condition turns OFF. For SET and RSET, the operand bit turns ON or OFF, respectively, when the input condition turns ON and the operand bit does not change when the input condition turns OFF.

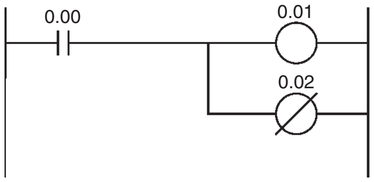
Precautions

- With CJ2 CPU Units, OUT can be used for bits in the DM or EM Area. For other models of CPU Unit, OUT cannot be used for bits in the DM or EM Area. Use OUTB(534) instead.
- If an indirect register address is used, OUT is executed even when the input condition turns OFF. Be particularly careful when programming OUT using an indirect index register address.



When the input condition is OFF, MOVR(560) is not executed, but OUT is executed for the address stored in the index register.

Example Programming



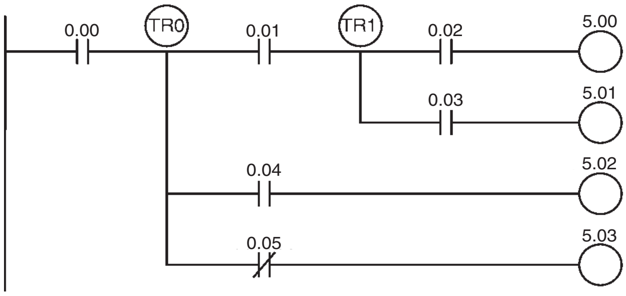
Instruction	Operand
LD	0.00
OUT	0.01
OUT NOT	0.02

TR

Instruction	Mnemonic	Variations	Function code	Function
TR Bits	TR	---	---	TR bits are used to temporarily retain the ON/OFF status of execution conditions in a program when programming in mnemonic code.

Function

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 Peripheral Device. The following diagram shows a simple application using two TR bits.



Coding

Instruction	Operands
LD	0.00
OUT	TR0
AND	0.01
OUT	TR1
AND	0.02
OUT	5.00
LD	TR1
AND	0.03
OUT	5.01
LD	TR0
AND	0.04
OUT	5.02
LD	TR0
AND NOT	5.00
OUT	5.03

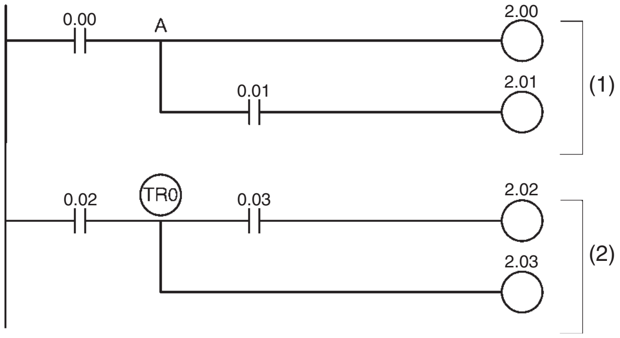
Relay Address

Temporary Relay	TR0 to TR15

● 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 2.00, so AND 0.01 and OUT 2.01 can be coded without requiring a TR bit. In instruction block (2), the status of the branching point and that of output CIO 2.02 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).

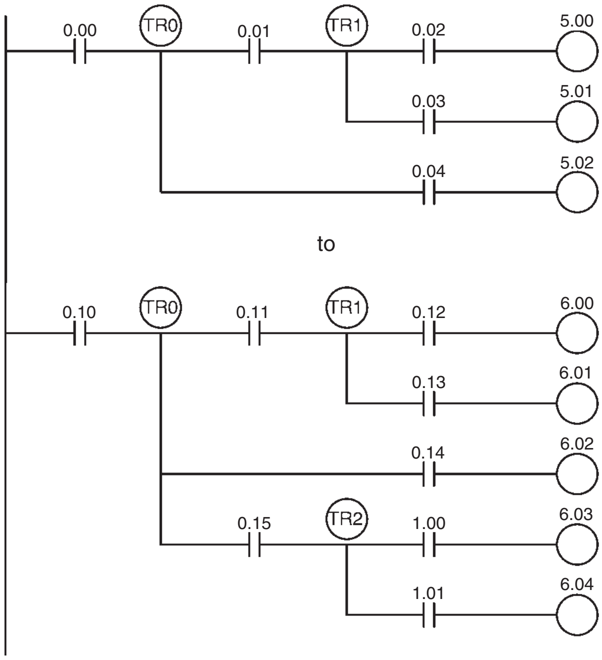


TR0 to TR15 Considerations

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.

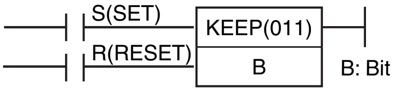
TR0 to TR15 output Duplication

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.



KEEP

Instruction	Mnemonic	Variations	Function code	Function
KEEP	KEEP	!KEEP	011	Operates like a latching relay.

Symbol	KEEP
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
B	Bit	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	---	---	OK*1	OK*1	---	---	---	---	---	OK	---	---	---	---

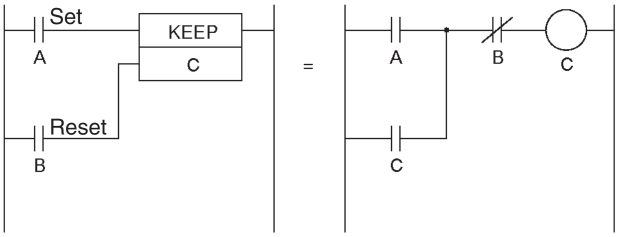
*1 CJ2 CPU Units only.

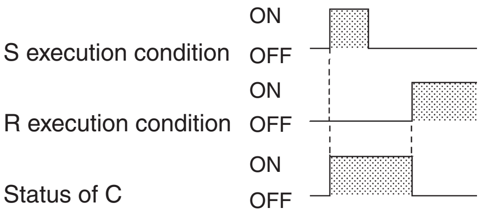
Flags

No flags are affected by KEEP(011).

Function

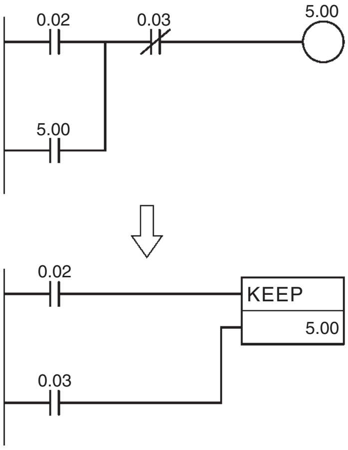
When S turns ON, the designated bit will go ON and stay ON until reset, regardless of whether S stays ON or goes OFF. When R turns ON, the designated bit will go OFF. The relationship between execution conditions and KEEP(011) bit status is shown below on the right.



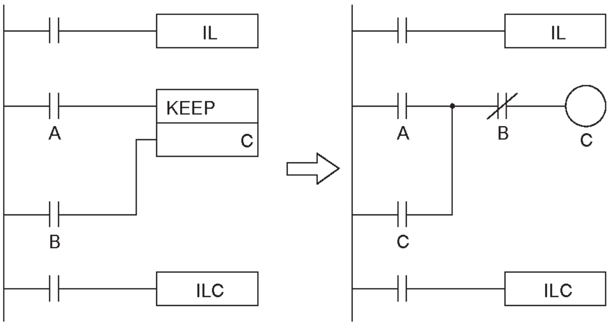


Hint

- 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. (The changes will not be reflected immediately if the bit is allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.)
- KEEP(011) operates like the self-maintaining bit, but a self-maintaining bit programmed with KEEP(011) requires one less instruction.



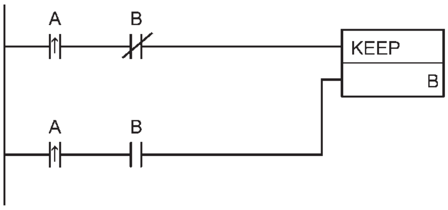
Self-maintaining bits programmed with KEEP(011) will maintain status even in an interlock program section, unlike the self-maintaining bit programmed without KEEP(011).

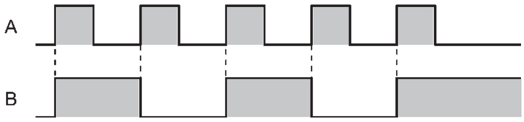


Output bit C will maintain its previous status in an interlock.

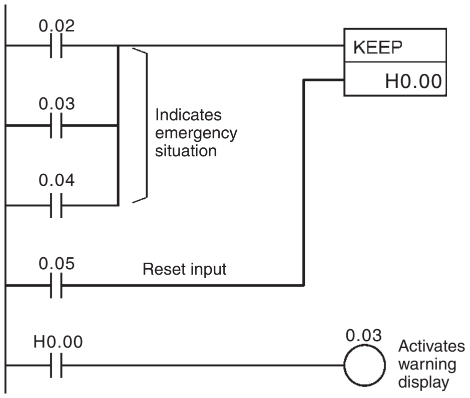
Output bit C will be turned OFF in an interlock.

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





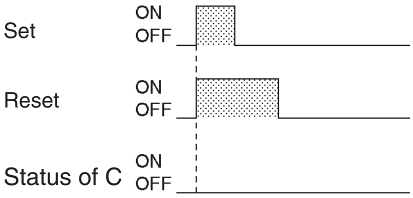
- If a holding bit is used for B, the bit status will be retained even during a power interruption. KEEP(011) can thus be used to program bits that will maintain status after restarting the PLC following a power interruption. An example of this that can be used to produce a warning display following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



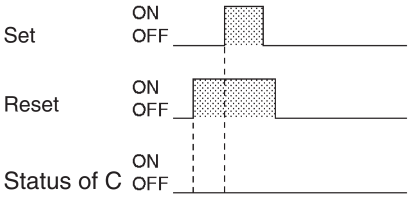
- The status of I/O Area bits can be retained in the event of a power interruption by turning ON the IOM Hold Bit and setting IOM Hold Bit Hold in the PLC Setup. In this case, I/O Area bits used in KEEP(011) will maintain status after restarting the PLC following a power interruption, just like holding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

Precautions

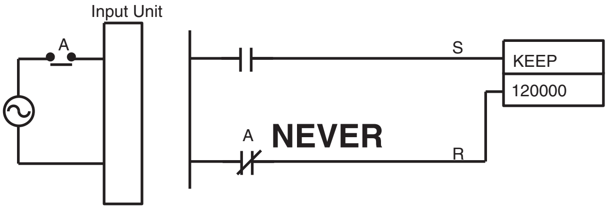
- If S and R are ON simultaneously, the reset input takes precedence.



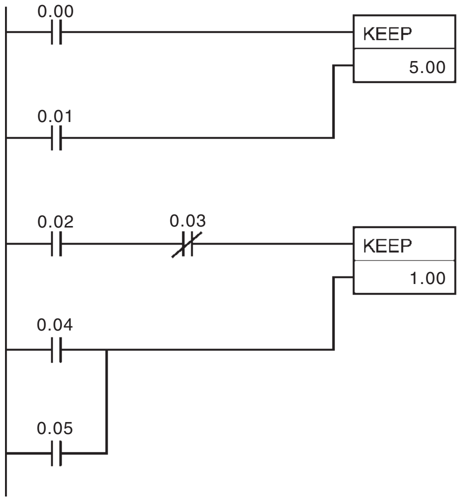
- The set input (S) cannot be received while R is ON.



- 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 on the right.



Example Programming



When CIO 0.00 goes ON in the left example, CIO 5.00 is turned ON. CIO 5.00 remains ON until CIO 0.01 goes ON.

When CIO 0.02 goes ON and CIO 0.03 goes OFF in the left example, CIO 1.00 is turned ON. CIO 1.00 remains ON until CIO 0.04 or CIO 0.05 goes ON.

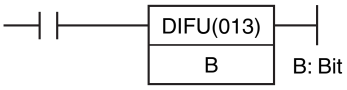
Coding

Instruction	Operand
LD	0.00
LD	0.01
KEEP (011)	5.00
LD	0.02
AND NOT	0.03
LD	0.04
OR	0.05
KEEP (011)	1.00

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

DIFU

Instruction	Mnemonic	Variations	Function code	Function
DIFFERENTIATE UP	DIFU	!DIFU	013	DIFU(013) turns the designated bit ON for one cycle when the execution condition goes from OFF to ON (rising edge).

	DIFU
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
B	Bit	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	---	---	OK*1	OK*1	---	---	---	---	---	OK	---	---	---	---

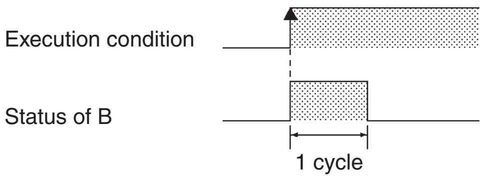
*1 CJ2 CPU Units only.

Flags

No flags are affected by DIFU(013).

Function

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.



Hint

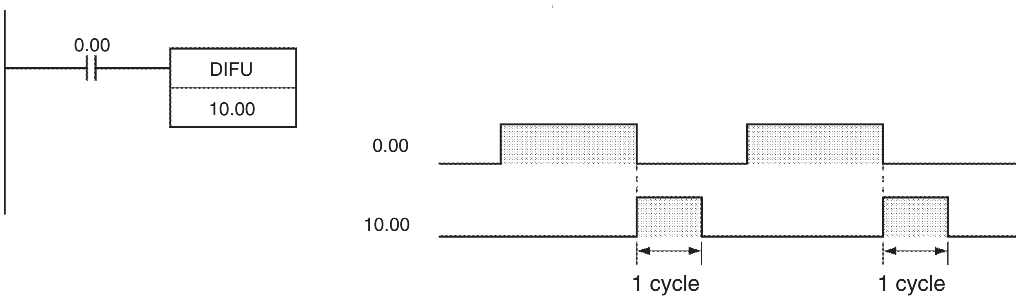
- UP(521) can be used to execute an instruction for just one cycle when the execution condition goes from OFF → ON.
- DIFU(013) has immediate refreshing variation (!DIFU(013)). When an external output bit has been specified for B in this instruction, any changes to B will be refreshed when the instruction is executed and reflected immediately in the output bit. (The changes will not be reflected immediately if the bit is allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.)

Precautions

- The operation of DIFU(013) 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.
- An instance will not be executed while EN is OFF. Caution is thus required when using DIFU(013) in a function block definition. For details, refer to information on restrictions on using ladder programming instructions in the CX-Programmer Operation Manual: Function Blocks.
- The operation of DIFU(013) will not be consistent if the same function block instance is executed more than once in the same cycle.
- An subroutine will not be executed while the input condition for the subroutine is OFF. Caution is thus required when using DIFU(013) in a function block definition. For details, refer to information on SBS(091).
- The operation of DIFU(013) will not be consistent if the same subroutine is executed more than once in the same cycle.

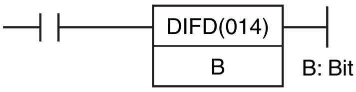
Example Programming

When CIO 0.00 goes from ON to OFF in the following example, CIO 10.00 is turned ON for one cycle.



DIFD

Instruction	Mnemonic	Variations	Function code	Function
DIFFERENTIATE DOWN	DIFD	!DIFD	014	DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge).

	DIFD
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
B	Bit	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	---	---	OK*1	OK*1	---	---	---	---	---	OK	---	---	---	---

*1 CJ2 CPU Units only.

Flags

No flags are affected by DIFD(014).

Function

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.

Hint

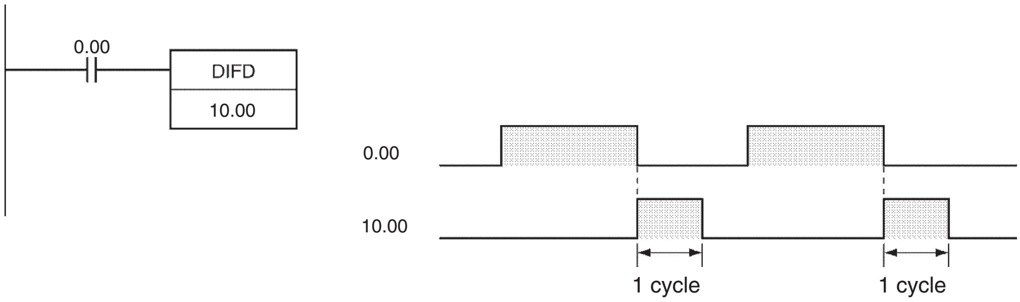
- DOWN(522) can be used to execute an instruction for just one cycle when the execution condition goes from ON → OFF.
- The operation of DIFD(014) depends on the execution condition for the instruction itself as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped program section, or a subroutine.
- DIFD(014) has immediate refreshing variation (!DIFD(014)). When an external output bit has been specified for B in this instruction, any changes to B will be refreshed when the instruction is executed and reflected immediately in the output bit. (The changes will not be reflected immediately if the bit is allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.)

Precautions

- The operation of 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.
- An instance will not be executed while EN is OFF. Caution is thus required when using DIFD(014) in a function block definition. For details, refer to information on restrictions on using ladder programming instructions in the CX-Programmer Operation Manual: Function Blocks.
- The operation of DIFD(014) will not be consistent if the same function block instance is executed more than once in the same cycle.
- An subroutine will not be executed while the input condition for the subroutine is OFF. Caution is thus required when using DIFD(014) in a function block definition. For details, refer to information on SBS(091).

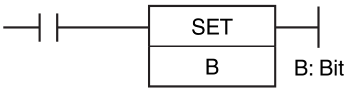
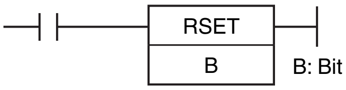
Example Programming

When CIO 0.00 goes from ON to OFF in the following example, CIO 10.00 is turned ON for one cycle.



SET/RSET

Instruction	Mnemonic	Variations	Function code	Function
SET	SET	@SET, %SET, !SET, !@SET, !%SET	---	SET turns the operand bit ON when the execution condition is ON. After this, the specified contact will remain ON regardless of ON/OFF of the input condition.
RESET	RSET	@RSET, %RSET, !RSET, !@RSET, !%RSET	---	RSET turns the operand bit OFF when the execution condition is ON. After this, the specified contact will remain OFF regardless of ON/OFF of the input condition.

Symbol	SET	RSET
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
B	Bit	BOOL	---

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	---	---	OK*1	OK*1	---	---	---	---	---	OK	---	---	---	---

*1 CJ2 CPU Units only.

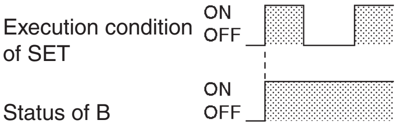
Flags

No flags are affected by SET and RSET.

Function

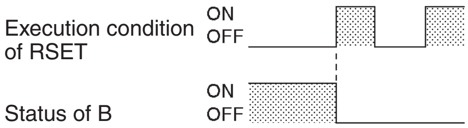
● SET

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

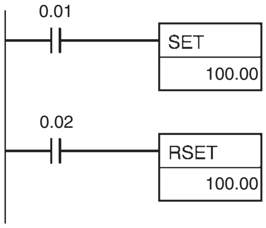
RSET turns the operand bit OFF when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF. Use SET to turn ON a bit that has been turned OFF with RSET.



Hint

- 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. For OUT, the operand bit is turned ON when the input condition turns ON and is turned OFF when the input condition turns OFF. For SET and RSET, the operand bit turns ON or OFF, respectively, when the input condition turns ON and the operand bit does not change when the input condition turns OFF.

CIO 100.00 is turned ON/OFF when CIO 0.00 goes ON/OFF.



CIO 100.00 is turned ON when CIO 0.01 goes ON; it remains ON until CIO 0.02 goes ON.

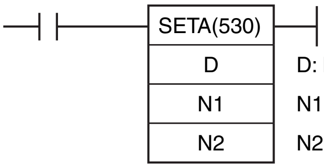
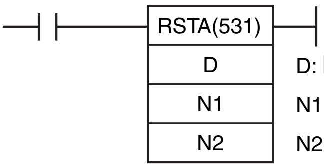
- The set and reset inputs for a KEEP(011) instruction must be programmed with the instruction, but the SET and RSET instructions can be programmed completely independently. Furthermore, the same bit may be used as the operand in any number of SET or RSET instructions.
- SET and RSET have immediate refreshing 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. (The changes will not be reflected immediately if the bit is allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.)
  If external output is specified for R by !SET (or !RSET), R will be OUT-refreshed as soon as it turns ON (or OFF) (when the instruction is executed). R, which turned ON (or OFF), will remain ON (or OFF) as normal until a RSET instruction (or SET instruction) is executed.

Precautions

- With CJ2 CPU Units, SET can be used for bits in the DM or EM Area. For other models of CPU Unit, SET cannot be used for bits in the DM or EM Area. Use SETB(532) instead. With CJ2 CPU Units, RSET can be used for bits in the DM or EM Area. For other models of CPU Unit, RSET cannot be used for bits in the DM or EM Area. Use RSTB(533) instead.
- 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.

SETA/RSTA

Instruction	Mnemonic	Variations	Function code	Function
MULTIPLE BIT SET	SETA	@SETA	530	SETA(530) turns ON the specified number of consecutive bits.
MULTIPLE BIT RESET	RESTA	@RSTA	531	RSTA(531) turns OFF the specified number of consecutive bits.

	SETA	RSTA
Symbol	 D: Beginning word N1: Beginning bit N2: Number of bits	 D: Beginning word N1: Beginning bit N2: Number of bits

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	Beginning Word	UINT	Variable
N1	Beginning Bit	UINT	1
N2	Number of Bits	UINT	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
N1											OK	OK						
N2																		

Flags

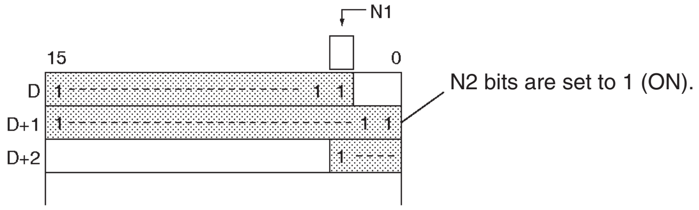
Operand	Description	Data type
Error Flag	P_ER	• ON if N1 is not within the specified range of 0000 to 000F (&0 to &15). • OFF in all other cases.

Function

● SETA

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).

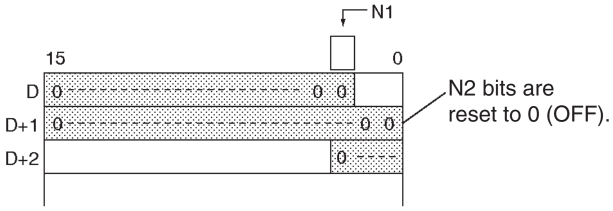


N2 bits are set to 1 (ON).

● RSTA

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).



Hint

● SETA

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

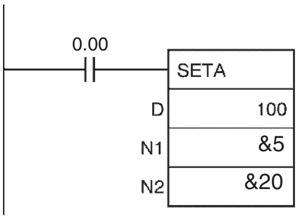
● RSTA

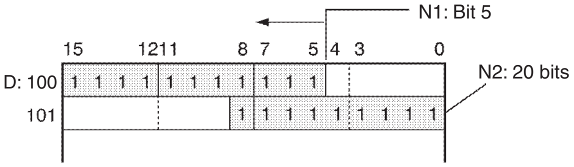
- RSTA(531) can be used to turn OFF bits in data areas that are normally accessed by words only, such as the DM and EM areas.

Example Programming

● SETA

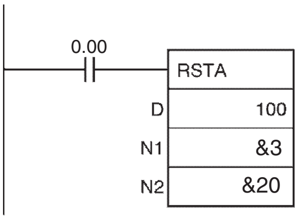
When CIO 0.00 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 5 of CIO 100 are turned ON.

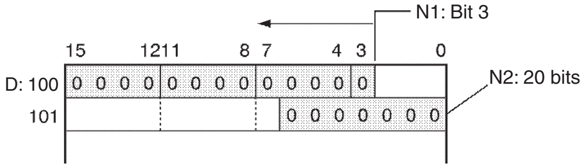




● RSTA

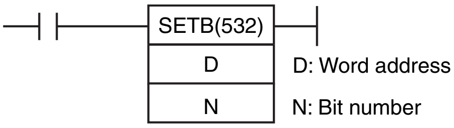
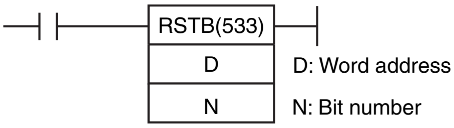
When CIO 0.00 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 3 of CIO 100 are turned OFF.





SETB/RSTB

Instruction	Mnemonic	Variations	Function code	Function
SINGLE BIT SET	SETB	@SETB, !SETB, !@SETB	532	SETB(532) turns ON the specified bit.
SINGLE BIT RESET	RSTB	@RSTB, !RSTB, !@RSTB	533	RSTB(533) turns OFF the specified bit.

	SETB	RSTB
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	Word address	UINT	1
N	Bit number	UINT	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
N											OK							

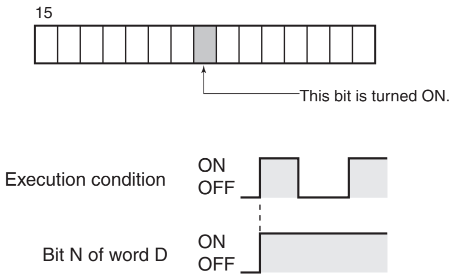
Flags

Operand	Description	Data type
Error Flag	P_ER	• ON if N is not within the specified range of 0000 to 000F (&0 to &15). • OFF in all other cases.

Function

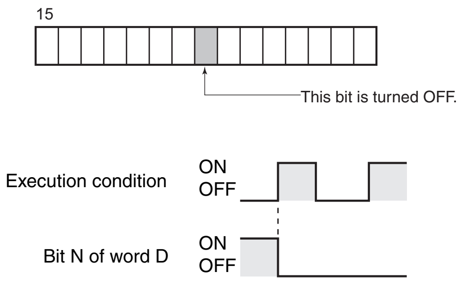
● SETB

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.



● RSTB

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.)



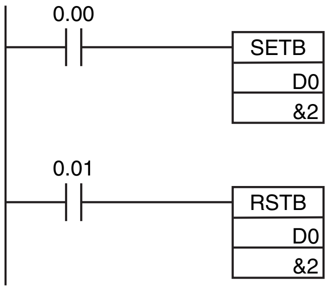
Hint

- Differences between SET/RSET and SETB(532)/RSTB(533)
  The instructions operate in the same way when the specified bit is in the CIO, W, H, or A Area.
  The SETB(532) and RSTB(533) instructions can control bits in the DM and EM Areas, unlike SET and RSET.
- Differences between OUTB(534) and SETB(532)/RSTB(533)
  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.
  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.
- 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.

Precautions

- Bits turned ON by SETB(532) can be turned OFF by any other instruction, not just RSTB(533).
  Bits turned OFF by RSTB(533) can be turned ON by any other instruction, not just SETB(532).
- SETB(532) and RSTB(533) cannot set/reset timers and counters.
- When SETB(532) or RSTB(533) is programmed between IL(002) and ILC(003) or JMP(004) and JME(005), the status of the specified bit will not be changed if the program section is interlocked or jumped, i.e., when the interlock condition or jump condition is OFF.
- SETB(532) and RSTB(533) have immediate refreshing variations (!SETB(532) and !RSTB(533)). When an external output bit has been specified in one of these instructions, any changes to the specified bit will be refreshed when the instruction is executed and reflected immediately in the output bit. (The changes will not be reflected immediately if the bit is allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.)
- When external output is specified for bit address N of word D by !SETB (or !RSTB instruction), bit address N of word D which turned ON (or OFF) will be OUT-refreshed at that point (when the instruction is executed). Bit address N of word D which was turned ON (or OFF) remains ON (or OFF) as normal until an RSTB instruction (or SETB instruction) is executed.

Example Programming

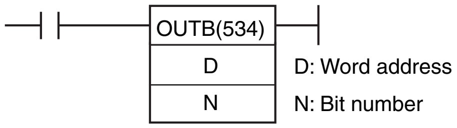


Bit 02 of D0 is turned ON when CIO 0.00 is ON.

Bit 02 of D0 is turned OFF when CIO 0.01 is ON.

OUTB

Instruction	Mnemonic	Variations	Function code	Function
SINGLE BIT OUTPUT	OUTB	!OUTB	534	OUTB(534) outputs the status of the instruction's execution condition to the specified bit.

	OUTB
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

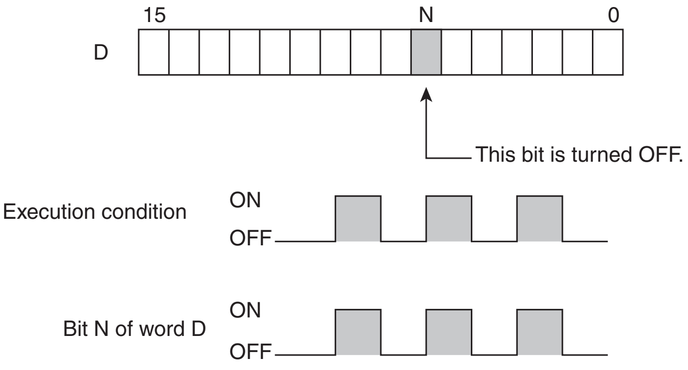
Operand	Description	Data type	Size
D	Word address	UINT	1
N	Bit number	UINT	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
N											OK							

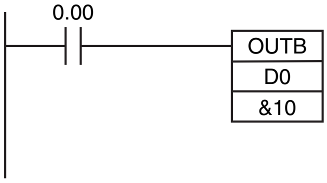
Function

When the execution condition is ON, OUTB(534) turns ON bit N of word D.
When the execution condition is OFF, OUTB(534) turns OFF bit N of word D.
If the immediate refreshing version is not used, the status of the execution condition (power flow) is written to the specified bit in I/O memory. If the immediate refreshing version is used, the status of the execution condition (power flow) is written to the Basic Output Unit's output terminal as well as the output bit in I/O memory.



Hint

- Difference between SETB(532)/RSTB(533) and OUTB(534)
  For OUTB(534), the operand bit is turned ON when the input condition turns ON and is turned OFF when the input condition turns OFF. For SETB(532) and RSTB(533), the operand bit turns ON or OFF, respectively, when the input condition turns ON and the operand bit does not change when the input condition turns OFF.

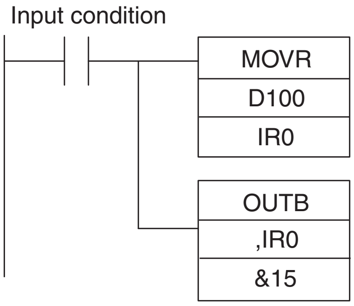


Bit 10 of D0 is turned OFF when CIO 0.00 is OFF.

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

Precautions

- 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.
- If an indirect index address is used, OUTB(534) is executed even when the input condition turns OFF. Be particularly careful when programming OUT using an indirect index register address.



When the input condition is OFF, MOVR(560) is not executed, but OUTB(534) is executed for the address stored in the index register.

Sequence Control Instructions

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.

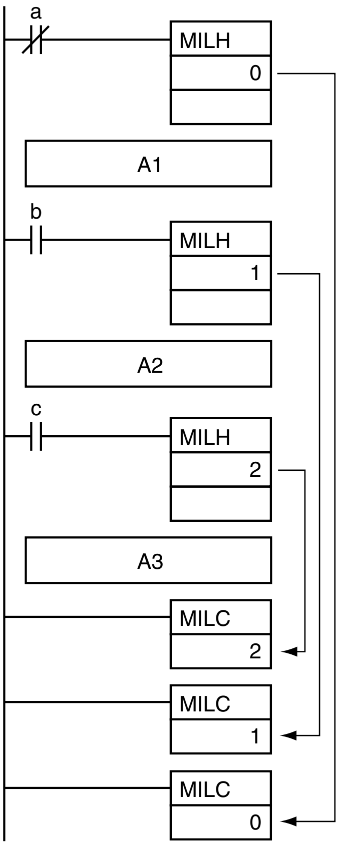
- MULTI-INTERLOCK DIFFERENTIATION RELEASE and MULTI-INTERLOCK CLEAR (MILR(518) and MILC(519))*

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.

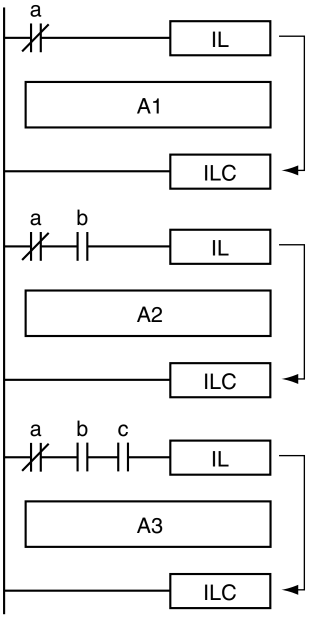
● Differences between Interlocks and Multiple Interlocks

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.

Interlocks with MILH and MILC



Interlocks with IL and ILC



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).

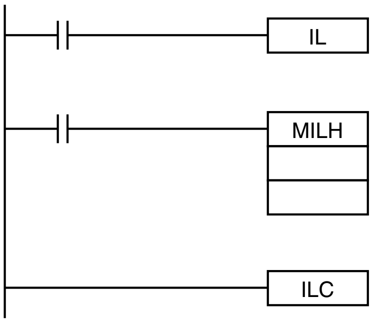
The operation of differentiated instructions in an interlock created with MILH(517) is identical to the operation in an interlock created with IL(002).

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

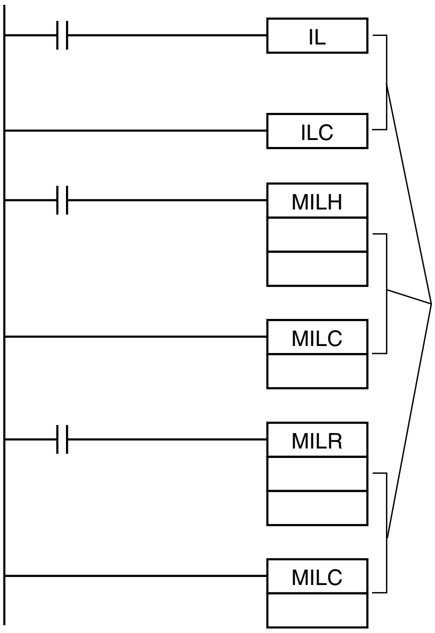
For example, an MILH(517) instruction cannot be inserted between IL(002) and IL(003).



MILH(517) is in an interlocked area between IL(002) and ILC.(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.



Different interlock methods can be used as long as the interlocked areas do not overlap.

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	Except OUT, OUT NOT, OUTB(534), and timer instructions, all 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), TIMU(541), TIMUX(556), TMUH(544), TMUHX(557) only) continue timing because the PVs are updated even when the timer instruction is not being executed.

END

Instruction	Mnemonic	Variations	Function code	Function
END	END	---	001	Indicates the end of a program.

Symbol	END
	 END(001)

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	Not allowed	Not allowed	OK	Not allowed

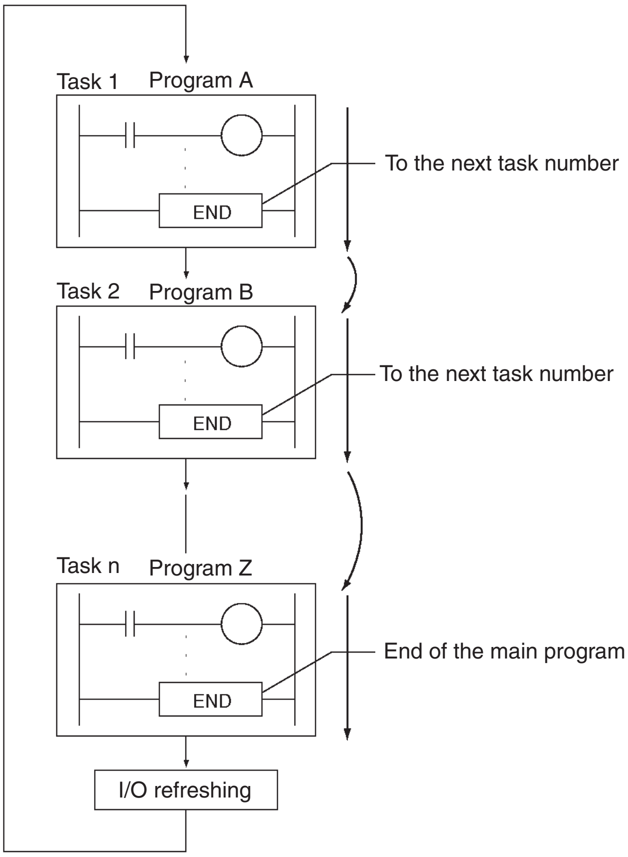
Flags

There are no flags affected by this instruction.

Function

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.

NOP

Instruction	Mnemonic	Variations	Function code	Function
NO OPERATION	NOP	---	000	This instruction has no function.

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

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

No flags are affected by NOP(000).

Function

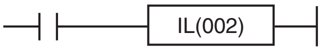
- No processing is performed for NOP(000), but this instruction can be used to set aside lines in the program where instructions will be inserted later.
- NOP(000) can only be used with mnemonic displays, not with ladder programs.

Hint

- When the instructions are inserted later, there will be no change in program addresses.

IL/ILC

Instruction	Mnemonic	Variations	Function code	Function
INTERLOCK	IL	---	002	Interlocks all outputs between IL(002) and ILC(003) when the execution condition for IL(002) is OFF.
INTERLOCK CLEAR	ILC	---	003	Indicates the end of the interlock range.

Symbol	IL	ILC
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	Not allowed	OK	OK	OK

Flags

Name	Label	Operation
Error Flag	ER	Unchanged *1
Equals Flag	=	Unchanged *1
Negative Flag	N	Unchanged *1

*1 In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D (for Single-CPU System) CPU Units, the Equals and Negative Flags are left unchanged.
In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

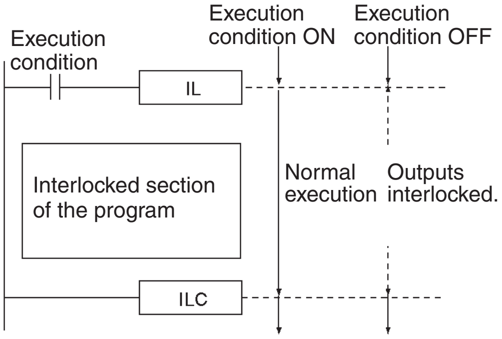
Function

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).

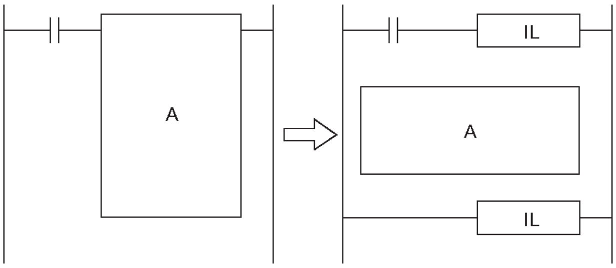
Instruction		Treatment
Bits specified in OUT, OUT NOT, or OUTB(534)		OFF
TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TIML(542), and TIMXL(553)	Completion Flag	OFF (reset)
	PV	Time set value (reset)
TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557)	Cannot be referenced.	
Bits/words specified 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.



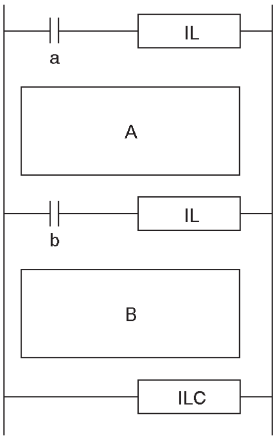
Hint

- 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).



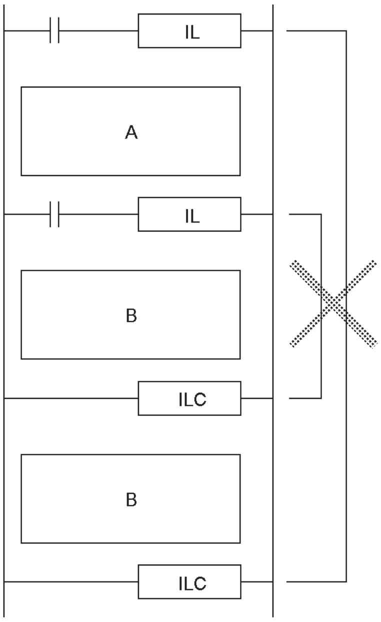
Precautions

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

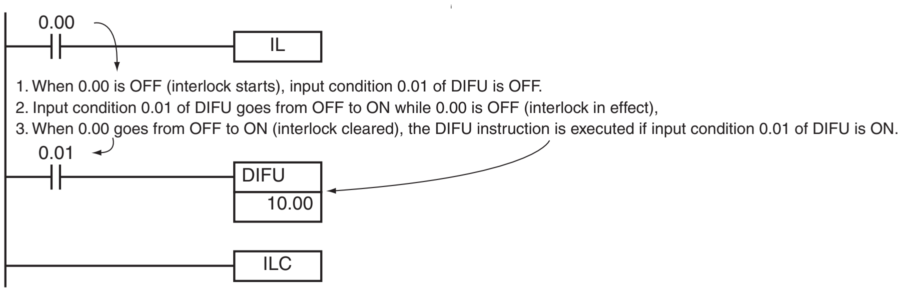
- IL(002) and ILC(003) cannot be nested, as in the following diagram. (Use MILH(517)/MILR(518) and MILC(519) when it is necessary to nest interlocks.)



● Operation of Differentiated Instructions

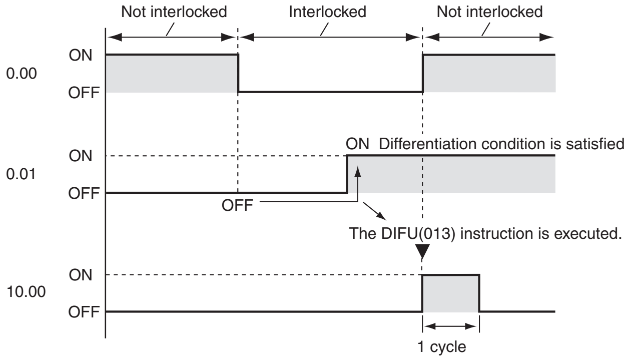
If there is a differentiated instruction (DIFU, DIFD, or instruction prefixed by @ or %) between IL(002) and ILC(003) instructions, that instruction will be executed when the interlock is cleared if the differentiation condition of the instruction is satisfied by means of a change in the input condition between starting and clearing of the interlock.

Example:
When a DIFFERENTIATE UP (DIFU(013)) instruction is used and the input condition is OFF when the interlock starts and ON when the interlock is cleared, the DIFFERENTIATE UP (DIFU(013)) instruction will be executed when the interlock is cleared.



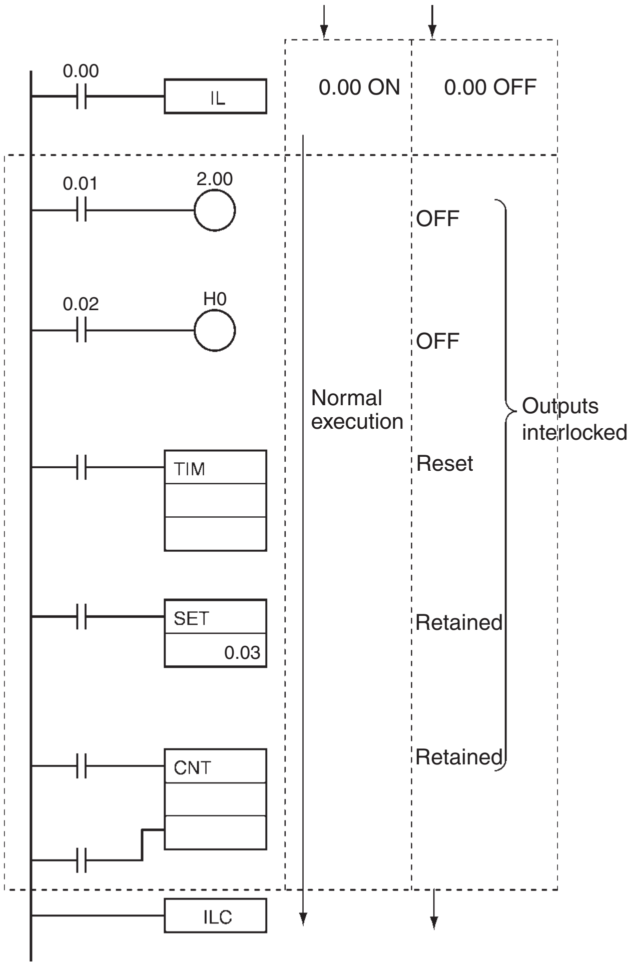
Reference:
An IL(002) instruction operates in the same way as an MILH(517) instruction in relation to differentiated instruction operation.

Timing Chart



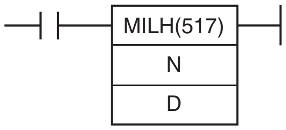
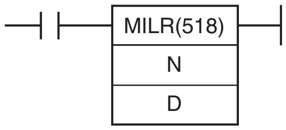
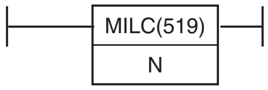
Example Programming

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



MILH/MILR/MILC

Instruction	Mnemonic	Variations	Function code	Function
MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	---	517	Interlocks all outputs between MILH(517) and MILC(519) when the execution condition for MILH(517) is OFF.
MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	---	518	Interlocks all outputs between MILR(518) and MILC(519) when the execution condition for MILR(518) is OFF.
MULTI-INTERLOCK CLEAR	MILC	---	519	Indicates the end of a multi-interlock range by means of an MILH or MILR instruction with the same interlock number.

	MILH	MILR	MILC
Symbol	 N: Interlock Number D: Interlock Status Bit	 N: Interlock Number D: Interlock Status Bit	 N: Interlock Number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	Not allowed	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Interlock number	--	1
D	Interlock status bit	BOOL	--

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
D	OK	OK	OK	OK							---			OK				

Flags

Name	Label	Operation
Error Flag	P_ER	OFF

Function

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.

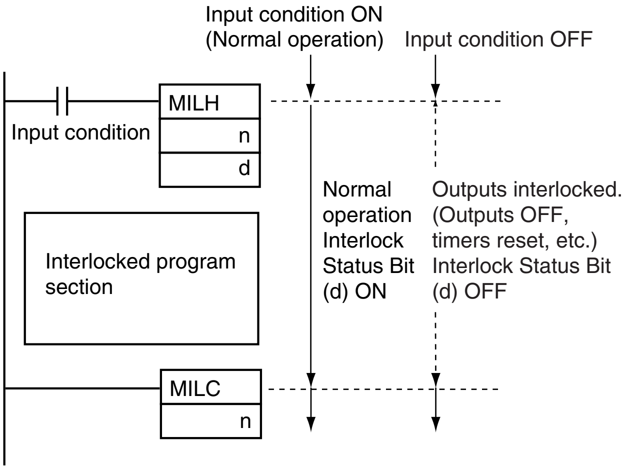
● Interlock Status

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)
TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557)	Cannot be referenced.	
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.



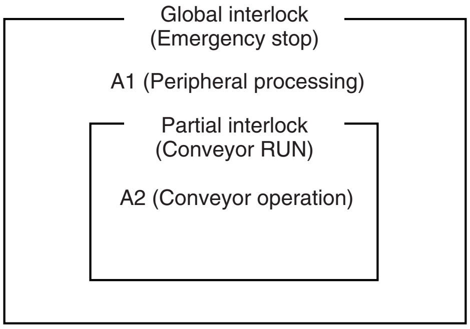
● Nesting

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.

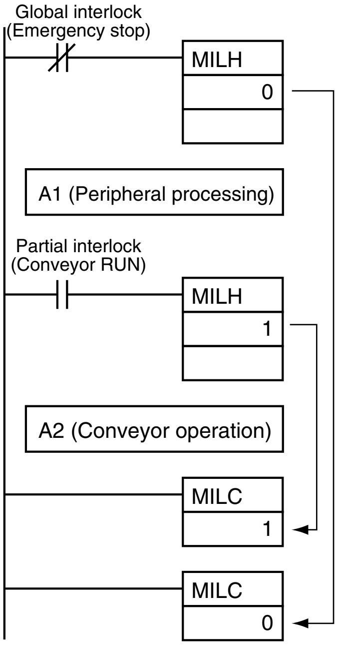
Nesting can be used for the following kinds of applications.

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.

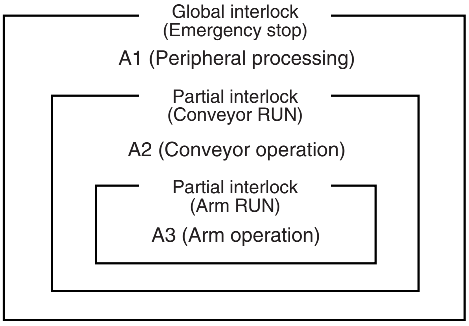


When the Emergency Stop is ON (input condition OFF), both A1 and A2 are interlocked.
When the Emergency Stop is OFF (input condition ON), A1 is executed normally and A2 is controlled by the Conveyor RUN switch as described below.

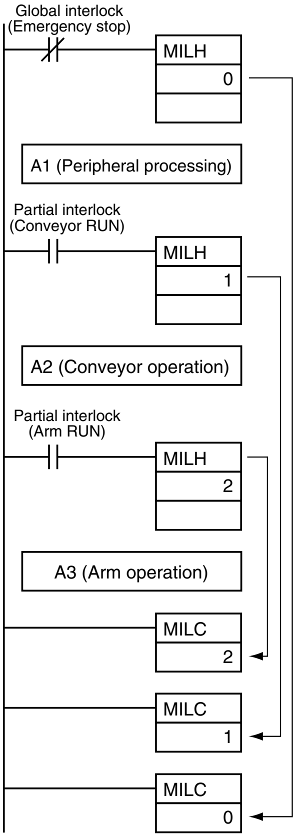
When the Conveyor RUN switch is OFF (input condition OFF), A2 is interlocked.
When the Conveyor RUN switch is ON (input condition ON), A2 is executed normally.

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.



When the Emergency Stop is ON (input condition OFF), A1, A2, and A3 are interlocked.
When the Emergency Stop is OFF (input condition ON), A1 is executed normally and A2 and A3 are controlled by the Conveyor RUN and Arm RUN switches as described below.

When the Conveyor RUN switch is OFF (input condition OFF), both A2 and A3 are interlocked.
When the Conveyor RUN switch is ON (input condition ON), A2 is executed normally and A3 is controlled by the Arm RUN switch as described below.

When the Arm RUN switch is OFF (input condition OFF), A3 is interlocked.
When the Arm RUN switch is ON (input condition ON), A3 is executed normally.

Differences between MILH(517) and MILR(518)

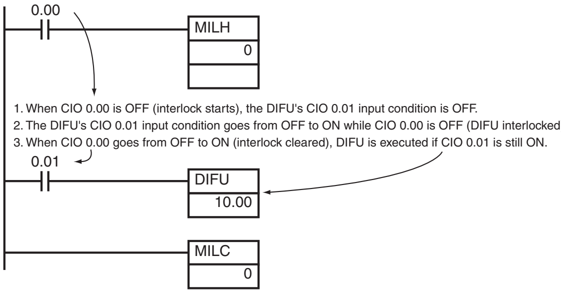
Differentiated instructions (DIFU, DIFD, or instructions with a @ or % prefix) operate differently in interlocks created with MILH(517) and MILR(518).
When a program section is interlocked with MILR(518), a differentiated instruction will not be executed when the interlock is cleared even if the differentiation 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.

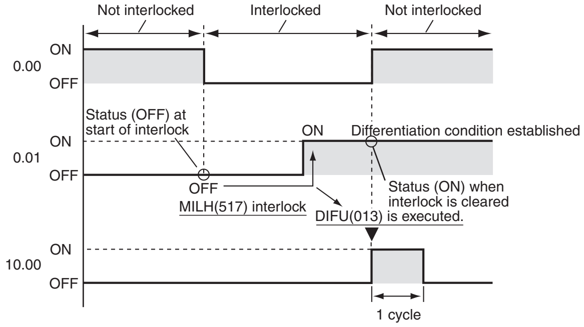
Operation of Differentiated Instructions in an MILH(517) Interlock

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.
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.)



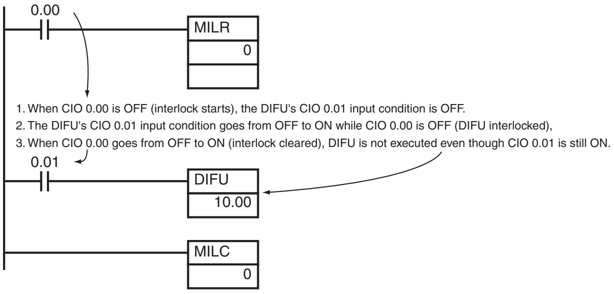
Timing chart



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

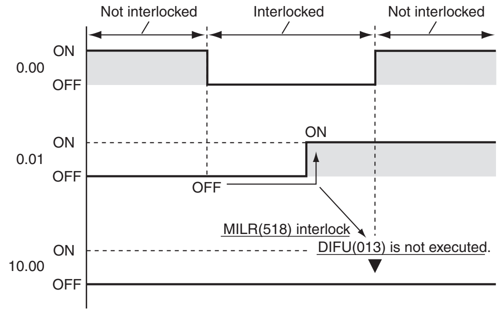
If there is a differentiated instruction (DIFU, DIFD, or instruction with a @ or % prefix) between MILR(518) and the corresponding MILC(519), that instruction will not be executed after the interlock is cleared even if the differentiation condition of the instruction was established.
In the same way, 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.



1. When CIO 0.00 is OFF (interlock starts), the DIFU's CIO 0.01 input condition is OFF.
2. The DIFU's CIO 0.01 input condition goes from OFF to ON while CIO 0.00 is OFF (DIFU interlocked),
3. When CIO 0.00 goes from OFF to ON (interlock cleared), DIFU is not executed even though CIO 0.01 is still ON.

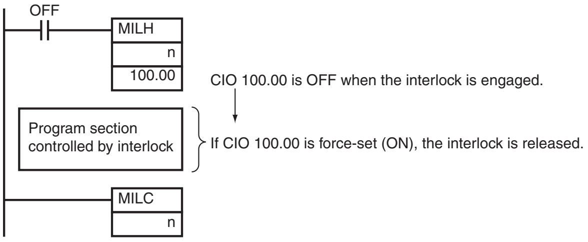
Timing chart



● Controlling Interlock Status from a Programming Device

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 a Programming Device. The forced status of the Interlock Status Bit has priority and overrides the interlock status calculated by program execution.

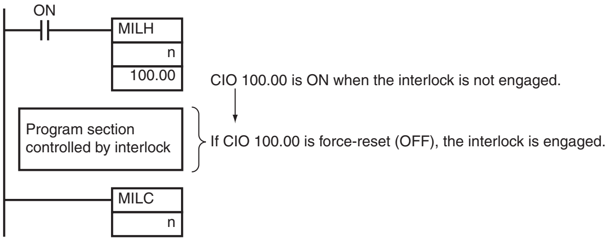
Force-set: Releases the interlock.



CIO 100.00 is OFF when the interlock is engaged.

If CIO 100.00 is force-set (ON), the interlock is released.

Force-reset: Engages the interlock.

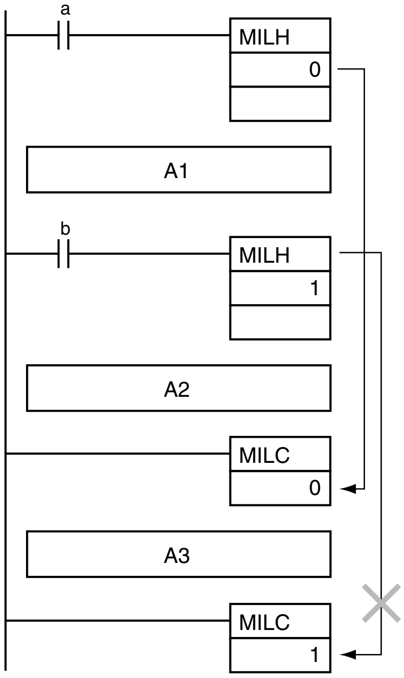


CIO 100.00 is ON when the interlock is not engaged.

If CIO 100.00 is force-reset (OFF), the interlock is engaged.

Hint

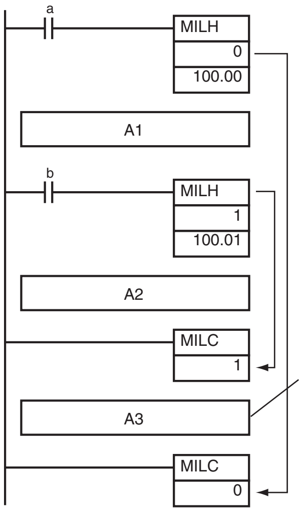
- The cycle time is not shortened when a section of the program is interlocked by MILH(517) or MILR(518) because the interlocked instructions are executed internally.
- When nesting interlocks, assign interlock numbers so that the nested program section does not exceed the outer program section.



The nested program section must not go beyond the outer program section.

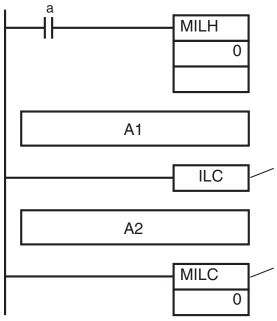
Execution condition		Program section		
a	**b**	**A1**	**A2**	**A3**
OFF	ON	Interlocked	Interlocked	Not interlocked
	OFF			
ON	OFF	Not interlocked	Interlocked	Interlocked
	ON	Not interlocked	Not interlocked	Not interlocked

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



Other instructions can be inserted between two MILC(519) instructions. In this case, sections A1 and A3 operate together. (They are interlocked when "a" is OFF, regardless of the ON/OFF status of "b".)

- 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.

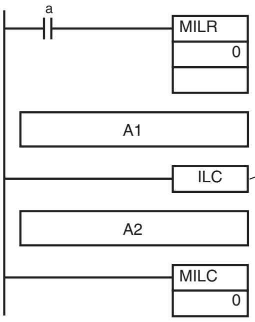


When input condition "a" is OFF, only program section A1 is interlocked.

If there is an ILC(003) instruction, the interlock is cleared at that point.

The MILC(519) instruction is ignored.

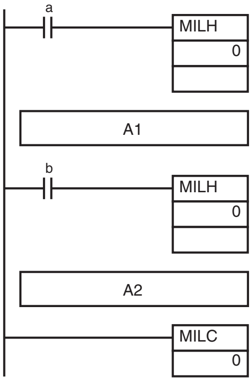
- If there is an ILC(003) instruction between an MILR(518) and MILC(519) pair, the ILC(003) instruction will be ignored and the full program section between MILR(518) and MILC(519) will be interlocked.



When input condition "a" is OFF, program sections A1 and A2 are interlocked.

The ILC(003) instruction is ignored.

- 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.

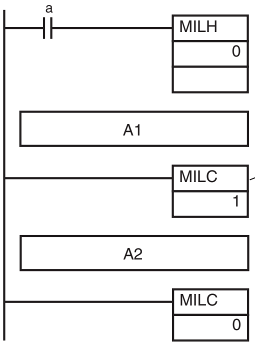


When input condition "a" is OFF, program sections A1 and A2 are both interlocked, even if input condition "b" is ON.

When input condition "a" is ON and "b" is OFF, only program section A2 is interlocked.

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.

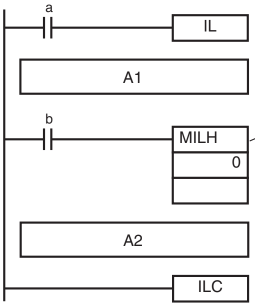
- If there is an MILC(519) instruction with a different interlock number between an MILH(517)/MILR(518) and MILC(519) pair, that MILC(519) instruction will be ignored.



When input condition "a" is OFF, program sections A1 and A2 are both interlocked.

This MILC(519) instruction is ignored.

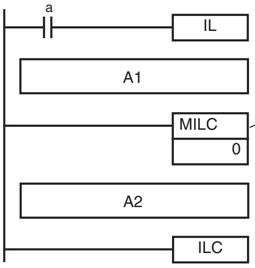
- If there is an MILH(517) instruction between an IL(002) and ILC(003) pair and the IL(002) interlock is engaged, the MILH(517) instruction has no effect. In this case, the program section between IL(002) and ILC(003) will be interlocked.
  If the IL(002) interlock is not engaged and the MILH(517) instruction's execution condition (b in this case) is OFF, the program section between MILH(517) and ILC(003) will be interlocked.



When input condition "a" is OFF, program sections A1 and A2 are both interlocked.

If the program section is not interlocked by IL(002) and "b" is OFF, program section A2 is interlocked.

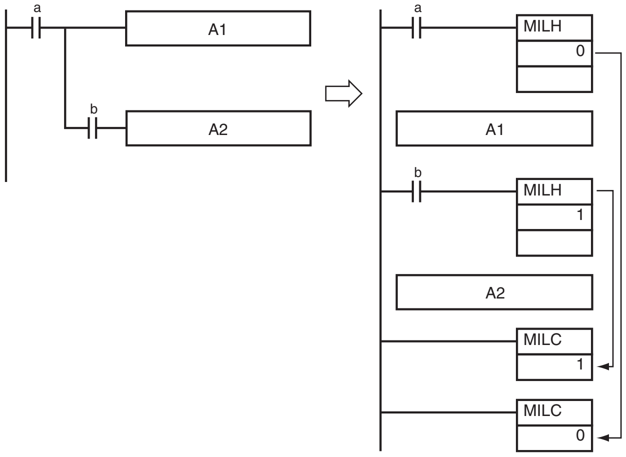
- If there is an MILC(519) instruction between an IL(002) and ILC(003) pair, that MILC(519) instruction will be ignored and the entire program section between IL(002) and ILC(003) will be interlocked.



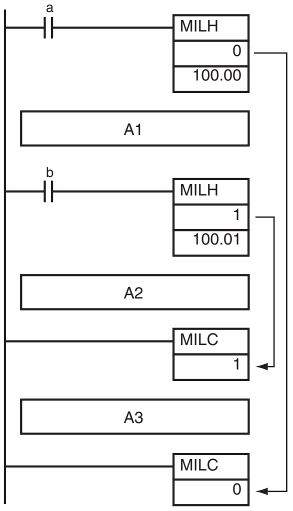
When input condition "a" is OFF, program sections A1 and A2 are both interlocked.

The MILC(519) instruction is ignored.

- 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.

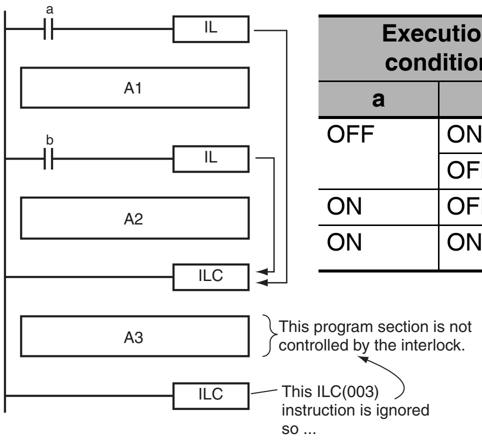


- Unlike the IL(002) interlocks, MILH(517) and MILR(518) interlocks can be nested, so the operation of similar programs will be different if MILH(517) or MILR(518) is used instead of ILC(002).
  - Program with MILH(517)/MILC(519) Interlocks



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	

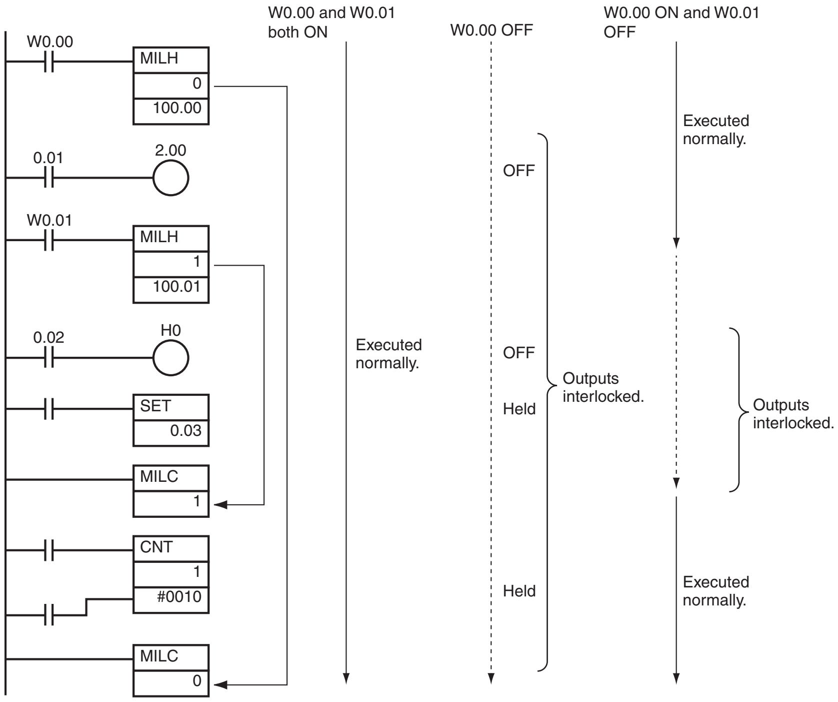
- If there are bits which you want to remain ON in a program section interlocked by MILH(517) or MILR(518), set these bits to ON with SET just before the MILH(517) or MILR(518) instruction.

Example Programming

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.

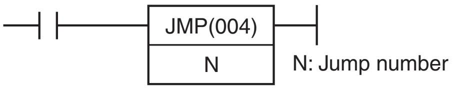
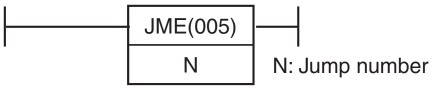
When W0.00 is OFF, the instructions between MILH(517) with interlock number 0 and MILC(519) with interlock number 0 are interlocked.

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.



JMP/JME

Instruction	Mnemonic	Variations	Function code	Function
JUMP	JMP	---	004	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.
JUMP END	JME	---	005	Indicates the end position of a jump by JMP or CJP instruction.

	JMP	JME
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	Not allowed	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
N	Jump number	UINT	1

N: Jump Number

The jump number must be 0000 to 03FF (&0 to &1,023 decimal).

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the jump number must be between the range 0000 to 00FF hex or &0 to &255 decimal.

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
JMP	N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK[1]	OK	---	OK	---	---	---	---
JME	N	---	---	---	---	---	---	---	---	---	---		---	---	---				

[1] For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is #0000 to #00FF (binary) or &0 to &1023 (decimal).

Flags

● JMP

Name	Label	Operation
Error Flag	ER	• ON if N is not within the specified range of 0000 to 03FF. (See note.) • 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.

*1 For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is 0 to 255 (0000 to 00FF hex).

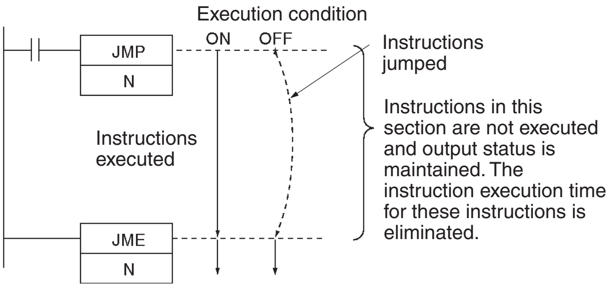
● JME

There are no flags affected by this instruction.

Function

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.



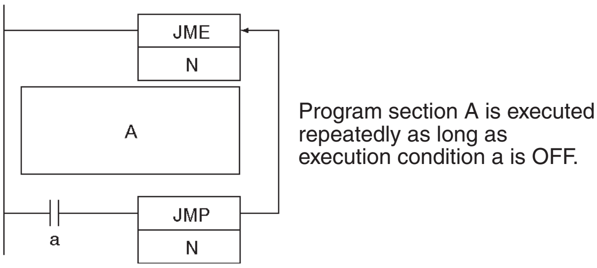
Hint

- Because all of instructions between JMP(004)/CJP(510)/CJPN(511) 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, processing time equivalent to NOP(000) processing is required 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.

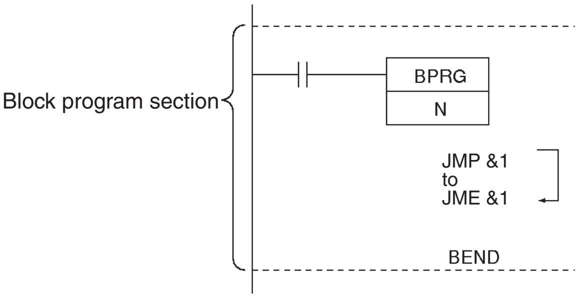
Item	JMP(004) JME(005)	CJP(510) JME(005)	CJPN(511) JME(005)	JMP0(515) JME0(516)
Execution condition for jump	OFF	ON	OFF	OFF
Number allowed	1,024 total (256 for CJ1M-CPU11/21.)			No limit
Instruction processing when jumped	Not executed.			NOP(000) processing
Instruction execution time when jumped	None			Equivalent to 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.

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), TMHHX(552), TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557)) continue timing because the PVs are updated even when the timer instruction is not being executed.
  Error will occur in the present value of any active TIMU(541), TIMUX(556), TMUH(544), TMUHX(557) instruction when the instruction is jumped.
- When there are two or more JME(005) instructions with the same jump number, only the instruction with the lower address will be valid. The JME(005) with the higher program address will be ignored.
- When JME(005) precedes JMP(004) in the program, the instructions between JME(005) and JMP(004) will be executed repeatedly as long as the execution condition for JMP(004) is OFF. A Cycle Time Too Long error will occur if the execution condition is not turned ON or END(001) is not executed within the maximum cycle time.



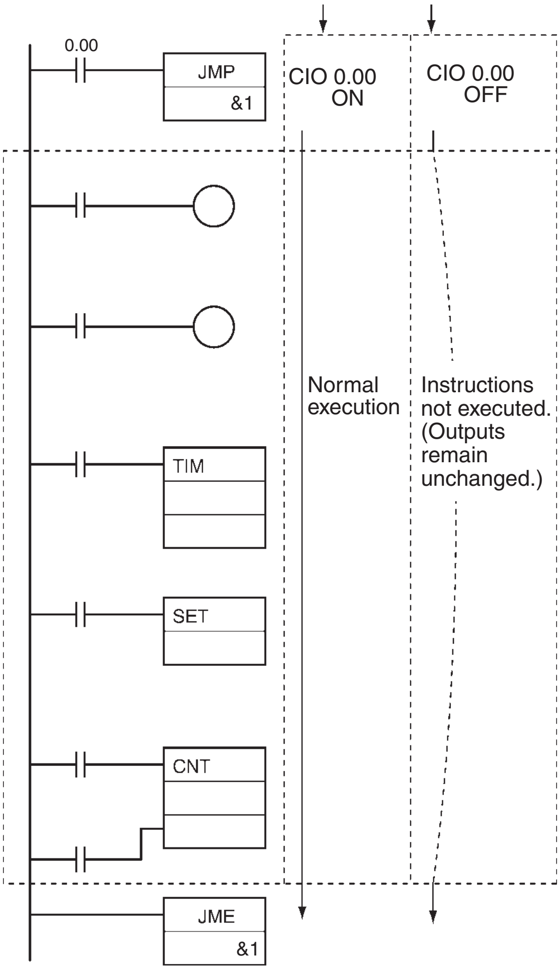
- In block programs, the instructions between JMP(004) and JME(005) are always skipped regardless of the status of the execution condition for JMP(004).



- JMP(004) and JME(005) pairs must be in the same task because jumps between tasks are not allowed. An error will occur if a JME(005) instruction is not programmed in the same task as its corresponding JMP(004) instruction.
- The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between JMP(004) and JME(005). When DIFU(013), DIFD(014), or a differentiated instruction is executed in an jumped section immediately after the execution condition for the JMP(004) has gone ON, the execution condition for the DIFU(013), DIFD(014), or differentiated instruction will be compared to the execution condition that existed before the jump became effective (i.e., before the execution condition for JMP(004) went OFF).

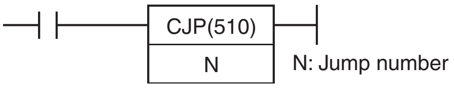
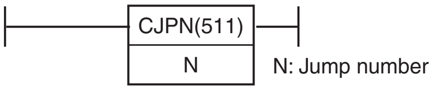
Example Programming

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



CJP/CJPN

Instruction	Mnemonic	Variations	Function code	Function
CONDITIONAL JUMP	CJP	---	510	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.
CONDITIONAL JUMP NOT	CJPN	---	511	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.

	CJP	CJPN
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	Not allowed	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
N	Jump number	UINT	1

N: Jump number

The jump number must be 0000 to 03FF (0 to 1,023 decimal).

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the jump number must be between the range 0000 to 00FF hex or &0 to &255 decimal.

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
CJP/ CJPN	N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK[*1]	OK	---	OK	---	---	---	---

[*1] For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is #0000 to #00FF (binary) or &0 to &1023 (decimal).

Flags

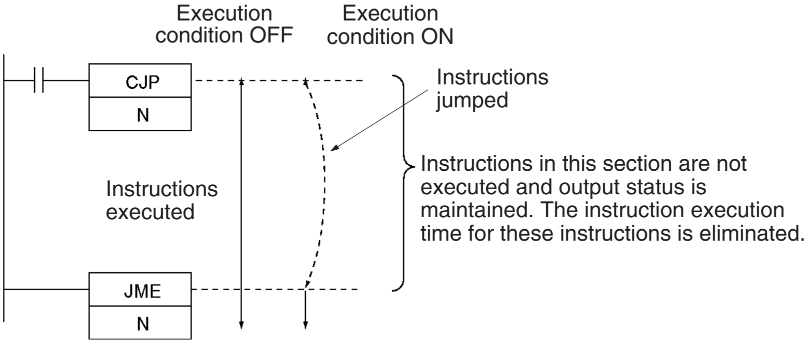
Name	Label	Operation
Error Flag	ER	• ON if there is not a JME(005) with the same jump number as CJP(510) or CJPN(511). (See note.) • ON if N is not within the specified range of 0000 to 03FF. • ON if there is a CJP(510) or CJPN(511) instruction in a task without a JME(005) with the same jump number. • OFF in all other cases.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the jump number must be between the range 0 to 255 (0000 to 00FF hex).

Function

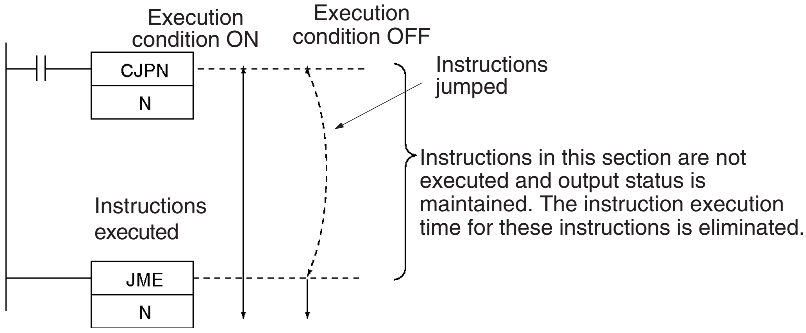
● CJP

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.



● CJPN

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.



Hint

- In the case of a JMP, CJP, or CJPN instruction, the program jumps directly to the JME instruction when the jump condition is satisfied. An instruction is not executed between JMP/CJP/CJPN and JME, and thus instruction execution time is not needed in that interval. Therefore, cycle time is reduced.
  By contrast, in the case of a JMP0 instruction, when the jump condition is satisfied, NOP processing (non-functional processing) is executed between JMP0 and JME0. During that interval, a processing time equivalent to a NOP instruction is required, and thus the cycle time will not be reduced.
- Functional comparison of jump instructions

Jump instruction	JMP-JME	CJP-JME	CJPN-JME	JMP0-JME0
Input condition for jump	OFF	ON	OFF	OFF
Number used	Total of 1024 (256 on the CJ1M-CPU11/21)			No restrictions
Instruction processing at jump	Non-execution			NOP processing
Execution time for jump	None			Total time is equivalent to a NOP instruction
Instruction output at jump	Holds the previous status			
Processing in block program area	Unconditional jump	Jump at ON	Jump at OFF	Not allowed

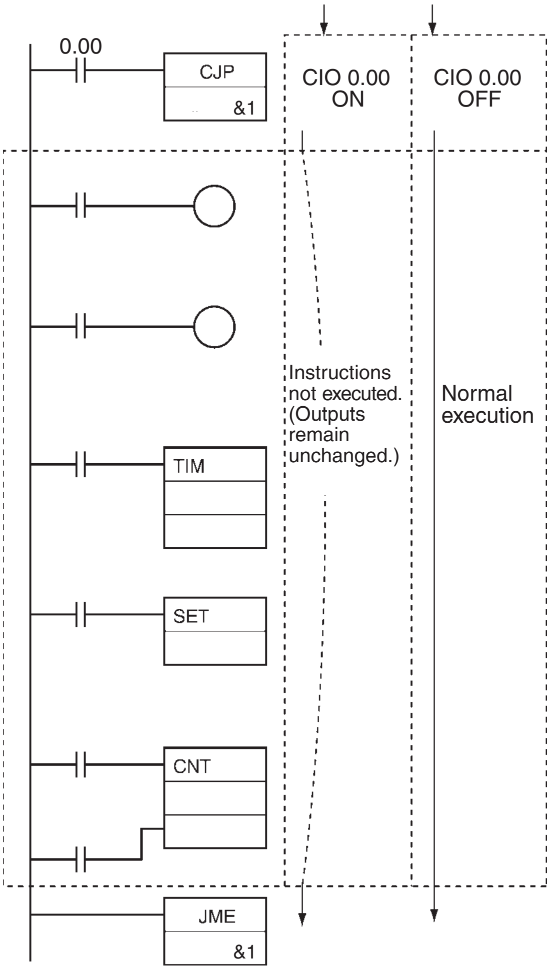
Precautions

- 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.
- 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.
  Error will occur in the present value of any active TIMU(541), TIMUX(556), TMUH(544), TMUHX(557) instruction when the instruction is jumped.
- 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.
- 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.
- In the block program area, the program will jump when the input condition prior to the CJP (CJPN) instruction is ON (OFF).
- 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 Programming

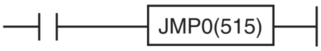
When CIO 0.00 is ON in the right example, the instructions between CJP(510) and JME(005) are not executed and the outputs maintain their previous status.
When CIO 0.00 is OFF in the right example, the instructions between CJP(510) and JME(005) are executed normally.



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

JMP0/JME0

Instruction	Mnemonic	Variations	Function code	Function
MULTIPLE JUMP	JMP0	---	515	When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000).
MULTIPLE JUMP END	JME0	---	516	

Symbol	JMP0	JME0
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	Not allowed	OK	OK	OK

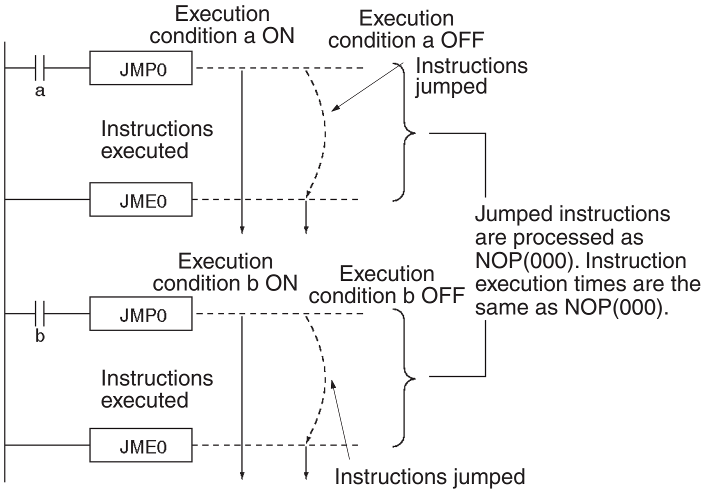
Flags

There are no flags affected by this instruction.

Function

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.



Hint

- In the case of a JMP, CJP, or CJPN instruction, the program jumps directly to the JME instruction when the jump condition is satisfied. An instruction is not executed between JMP/CJP/CJPN and JME, and thus instruction execution time is not needed in that interval. Therefore, cycle time is reduced.
  By contrast, in the case of a JMP0 instruction, when the jump condition is satisfied, NOP processing (non-functional processing) is executed between JMP0 and JME0. During that interval, a processing time equivalent to a NOP instruction is required, and thus the cycle time will not be reduced.
- Functional comparison of jump instructions

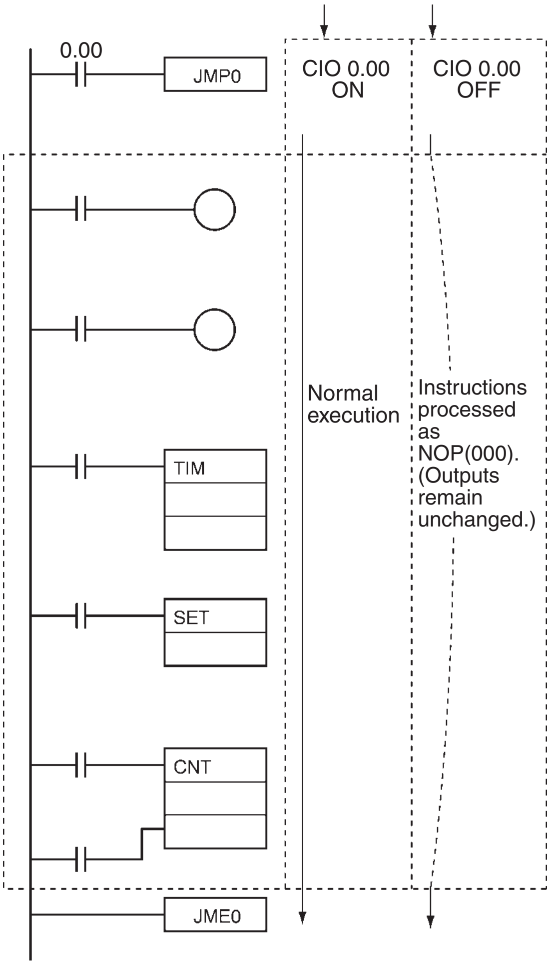
Jump instruction	**JMP-JME**	**CJP-JME**	**CJPN-JME**	**JMP0-JME0**
Input condition for jump	OFF	ON	OFF	OFF
Number used	Total of 1024 (256 on the CJ1M-CPU11/21)			No restrictions
Instruction processing at jump	Non-execution			NOP processing
Execution time for jump	None			Total time is equivalent to a NOP instruction
Instruction output at jump	Holds the previous status			
Processing in block program area	Unconditional jump	Jump at ON	Jump at OFF	Not allowed

Precautions

- 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.
- 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).

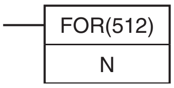
Example Programming

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.



FOR/NEXT

Instruction	Mnemonic	Variations	Function code	Function
---	FOR	---	512	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times.
	NEXT	---	513	

Symbol	FOR	NEXT
	 N: Number of loops	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Number of loops	UINT	1

N: Number of loops

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

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

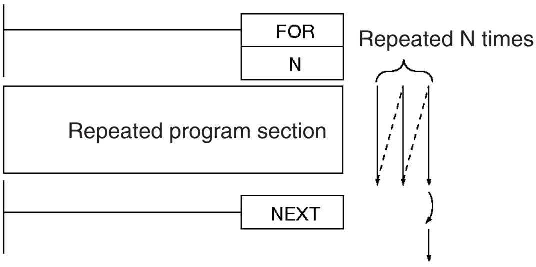
Name	Label	Operation
Error Flag	ER	• ON if more than 15 loops are nested. • OFF in all other cases.
Equals Flag	=	OFF
Negative Flag	N	OFF

Function

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.



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.

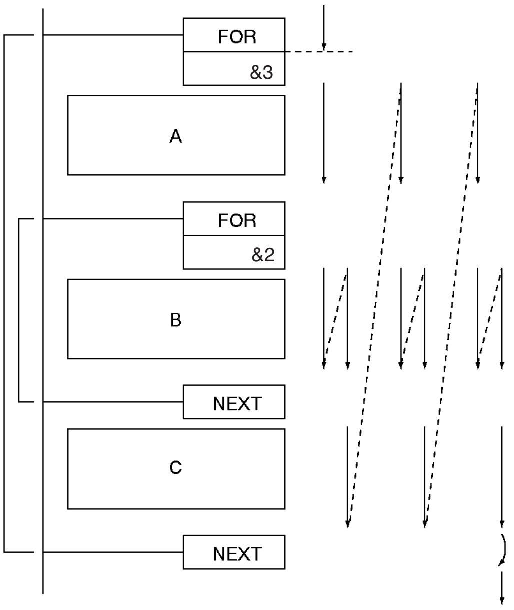
Hint

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

- 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.
- 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.)

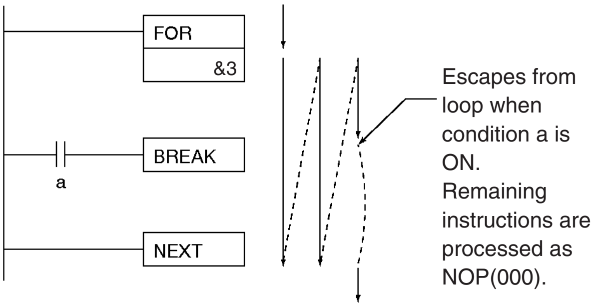
Precautions

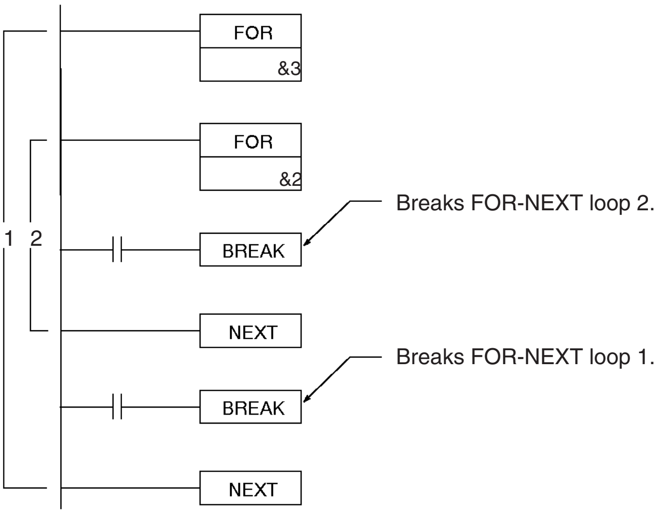
- Program FOR(512) and NEXT(513) in the same task. Execution will not be repeated if these instructions are not in the same task.
- FOR-NEXT loops can be nested up to 15 levels.



In the example above, program sections A, B, and C are executed as follows:
A → B → B → C, A → B → B → C, and A → B → B → C

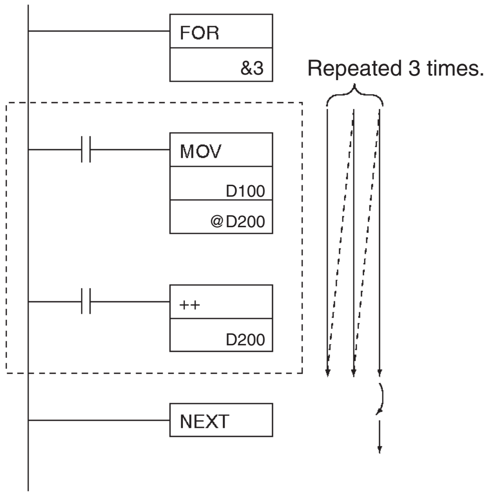
- Use BREAK(514) to escape from a FOR-NEXT loop. Several BREAK(514) instructions (the number of levels nested) are required to escape from nested loops. The remaining instructions in the loop after BREAK(514) are processed as NOP(000) instructions.



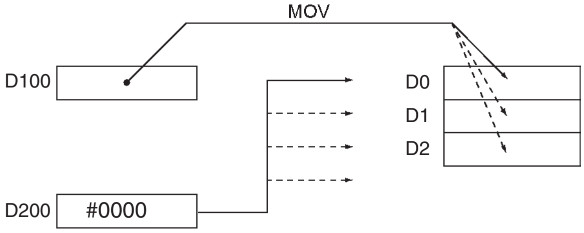


- A jump instruction such as JMP(004) may be executed within a FOR-NEXT loop, but do not jump beyond the FOR-NEXT loop.
- The following instructions cannot be used within FOR-NEXT loops:
  - Block programming instructions
  - MULTIPLE JUMP and JUMP END: JMP(515) and JME(516)
  - STEP DEFINE and STEP START: STEP(008)/SNXT(009)
- If the following differentiated instructions are used between FOR and NEXT, the differentiated instruction will be executed only once. It will not be executed for the number of loops.
  - UP and DOWN
  - DIFU and DIFD
  - Upwardly differentiated versions of instructions (instructions with @ option)
  - Downwardly differentiated versions of instructions (instructions with % option)

Example Programming

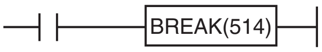


In the left example, the looped program section transfers the content of D100 to the address indicated in D200 and then increments the content of D200 by 1.



BREAK

Instruction	Mnemonic	Variations	Function code	Function
BREAK LOOP	BREAK	---	514	Programmed in a FOR-NEXT loop to cancel the execution of the loop for a given execution condition. The remaining instructions in the loop are processed as NOP(000) instructions.

	BREAK
Symbol	

Applicable Program Areas

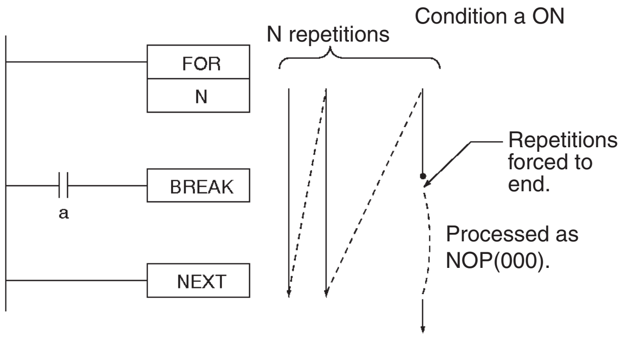
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Flags

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

Function

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).



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.

Timer and Counter Instructions

Timer Instructions

Refresh Methods for Timer/Counter PV

● Overview

There are two PV refresh methods for instructions related to timer/counters, “BCD” and “BINARY”.

Method	Description	Setting range	Set value
BCD	Sets the timer set value in BCD.	0~9.999	#0000~9999
Binary	Sets the timer set value in BINARY.	0~65.535	&0~65535 or #0000~FFFF

The PLC Setup for all of the timer/counter-related instructions. The refresh method is valid also 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.)

● Applicable Instructions

Classification	Instruction	Mnemonic	
		BCD	Binary
Timer/counter instructions	HUNDRED-MS TIMER	TIM	TIMX(550)
	TEN-MS TIMER	TIMH(015)	TIMHX(551)
	ONE-MS TIMER	TMHH(540)	TMHHX(552)
	TENTH-MS TIMER (See note.)	TIMU(541)	TIMUX(556)
	HUNDREDTH-MS TIMER (See note.)	TMUH(544)	TMUHX(557)
	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	HUNDRED-MS TIMER WAIT	TIMW(813)	TIMWX(816)
	TEN-MS TIMER WAIT	TMHW(815)	TMHWX(817)
	COUNTER WAIT	CNTW(814)	CNTWX(818)

● Setting method for PV refresh

- In the CS1-H, CS1D, CJ1-H, and CS1M CPU Units, either BCD or binary PV refreshing must be set in the PLC Setup for the entire product. Select the method by "Execute timer/counter in binary mode" from the PLC properties of CX-Programmer.
- In the CJ2 CPU Units, BCD and binary PV refreshing can both be used in the same project. The setting of the PV refresh method in the PLC Setup will be ignored.

Basic Timer Specifications

Item		TIM/TIMX (550)	TIMH(015)/ TIMHX(551)	TMHH(540)/ TMHHX(552)	TIMU(541)/ TIMUX(556) (See note 3.)	TMUH(544)/ TMUHX(557) (See note 3.)	TTIM(087)/ TTIMX(555)	TIML(542)/ TIMLX(553)	MTIM(543)/ MTIMX(554)
Timing method		Decrementing	Decrementing	Decrementing	Decrementing	Decrementing	Incrementing	Decrementing	Incrementing
Timing units		100 ms	10 ms	1 ms	0.1 ms	0.01 ms	100 ms	100 ms	100 ms
Maximum 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	TIMU: 0.9999 s TIMUX: 6.5535 s	TMUH: 0.09999 s TMUHX: 0.65535 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	1	1	8
Timer numbers		Used	Used	Used	Used	Used	Used	Not used	Not used
Completion Flag refreshing		When the instruction is executed	When the instruction is executed	When the instruction is executed	When the instruction is executed	When the instruction is executed	When the instruction is executed	When the instruction is executed	When the instruction is executed
Timer PV refreshing (See note)		TIM PVs are refreshed when the instruction is executed, after executing all cyclic tasks each cycle, or every 80 ms by interrupt if the cycle time exceeds 80 ms.	TIMH(015)/TIMHX(551) PVs are refreshed when the instruction is executed, after executing all cyclic tasks each cycle, and every 10 ms by interrupt.	• Every 1 ms • When the instruction is executed	• The PV cannot be read.	• The PV cannot be read.	When the instruction is executed	When the instruction is executed	When the instruction is executed
Value after reset	**Completion Flags**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
	PVs	SV	SV	SV	---		0	SV	0

Note Timers are refreshed at different times depending on the timer number. Refer to the descriptions of individual timer instructions for details

Item		TIM/TIMX (550)	TIMH(015)/ TIMHX(551)	TMHH(540)/ TMHHX(552)	TIMU(541)/ TIMUX(556)	TMUH(544)/ TMUHX(557)	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 Completion Flag = OFF	Timer status maintained.
Forced set	**Completion Flag**	ON						---	---
	PVs	Set to 0.			--- (See note.)		Set to 0.	---	---
Forced reset	**Completion Flags**	OFF						---	---
	PVs	Reset to SV.			--- (See note.)		Set to 0.	---	---

Note It is not possible to read the timer PVs of TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557).

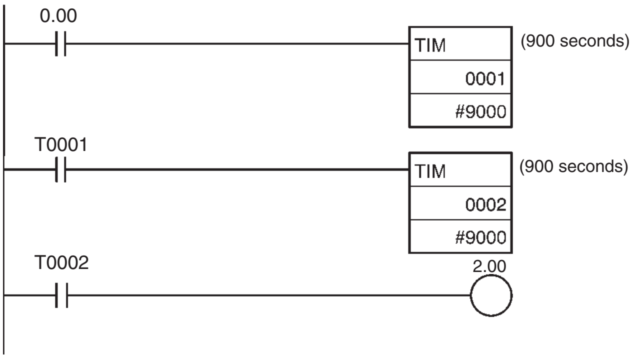
Example Timer and Counter Applications

Example 1: Long-term Timers

The following program examples show three ways to create long-term timers with standard TIM and CNT instructions.

1) Two TIM Instructions

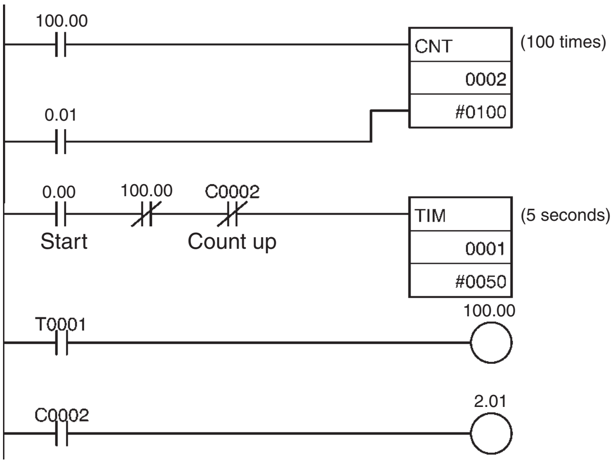
In this example, two TIM instructions are combined to make a 30-minute timer.



Instruction	Operands
LD	0.00
TM	1
	#9000
LD	T0001
TM	2
	#9000
LD	T0002
OUT	2.00

2) 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.

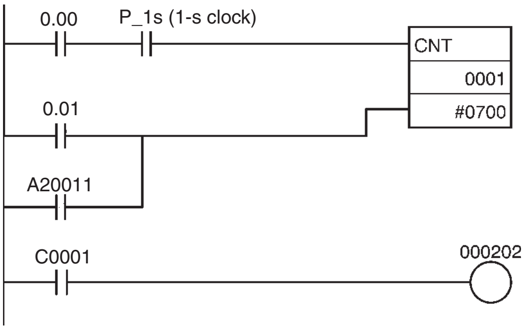


Instruction	Operands
LD	100.00
LD	0.01
CNT	2
	#0100
LD	0.00
AND NOT	100.00
AND NOT	C0002
TIM	1
	#0050
LD	T0001
OUT	100.00
LD	C0002
OUT	2.01

3) 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.

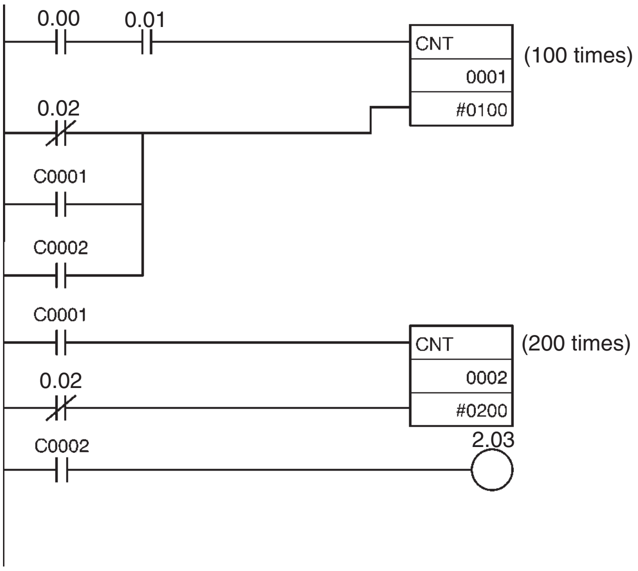
If the First Cycle Flag (A200.11) is ORed with the counter's reset input (CIO 0.01), the counter's PV will be reset to the SV (0700) when program execution begins rather than resuming the count from the previous PV.



Instruction	Operands
LD	0.00
AND	1s
LD NOT	0.01
CNT	1
	#0700
	C0001
	2.02

Example 2: Two-stage Counter

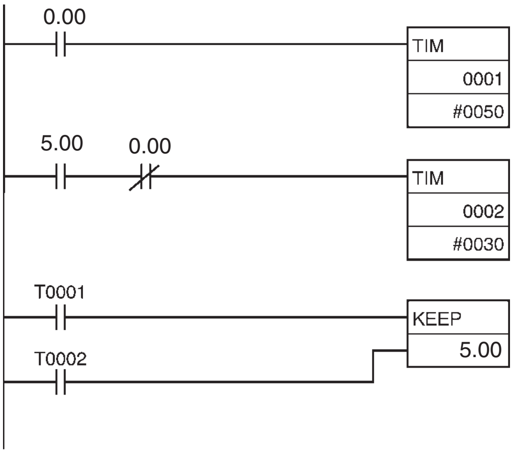
When an SV higher than 9999 is required, two counters can be combined as shown in the following example. In this case, two CNT instructions are combined to make a BCD counter with an SV of 20,000.



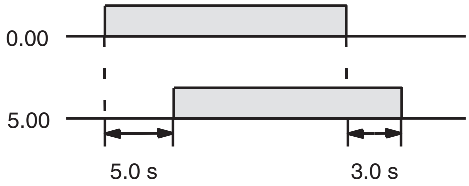
Instruction	Operands
LD	0.00
AND	0.01
LD NOT	0.02
OR	C0001
OR	C0002
CNT	1
	#0100
LD	C0001
LD NOT	0.02
CNT	2
	#0200
LD	C0002
OUT	2.03

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 5.00 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.

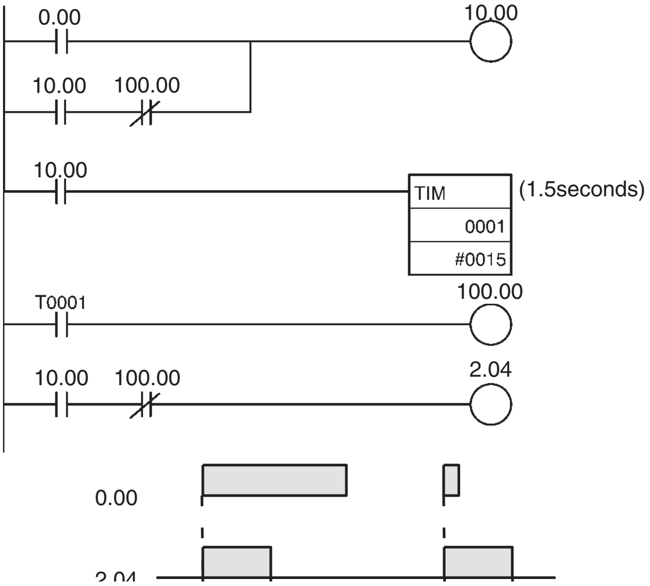


Instruction	Operands
LD	0.00
TIM	1
	#0050
LD	5.00
LD NOT	0.00
TIM	2
	#0030
LD	T0001
LD	T0002
KEEP(011)	5.00



Example 4: One-shot Bit

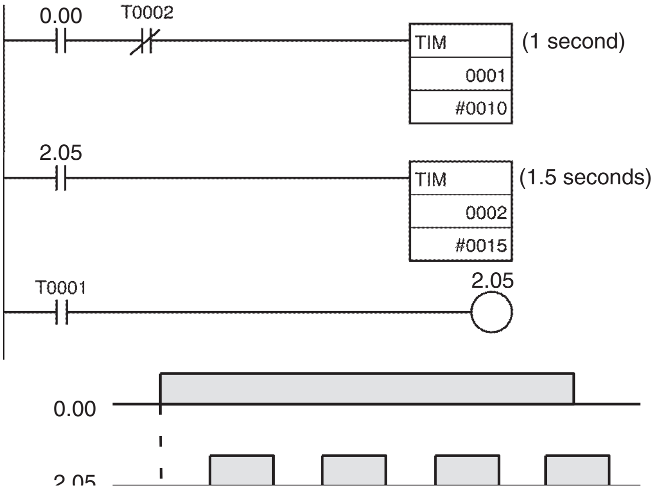
A TIM timer can be combined with OUT or OUT NOT to control how long a particular bit is ON or OFF. In this example, CIO 2.04 will be ON for 1.5 seconds (the SV of T0001) after CIO 0.00 goes ON.



Instruction	Operands
LD	0.00
LD	10.00
AND NOT	100.00
OR LD	--
OUT	10.00
LD	10.00
TIM	1
	#0015
LD	T0001
OUT	100.00
LD	10.00
AND NOT	100.00
OUT	2.04

Example 5: Flicker Bit
1) Two TIM Instructions

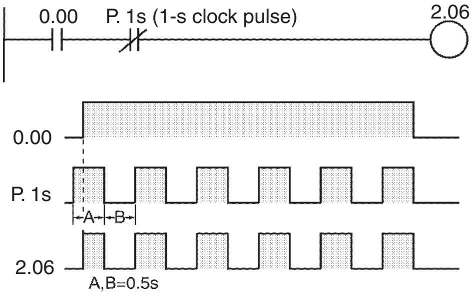
Two TIM timers can be combined to make a bit turn ON and OFF at regular intervals while the execution condition is ON. In this example, CIO 2.05 will be OFF for 1.0 second and then ON for 1.5 seconds as long as CIO 0.00 is ON.



Instruction	Operands
LD	0.00
AND LD	T0002
TIM	1
	#0010
LD	2.05
TIM	2
	#0015
LD	T0001
OUT	2.05

2) 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).



Instruction	Operands
LD	0.00
AND	1s
OUT	2.6

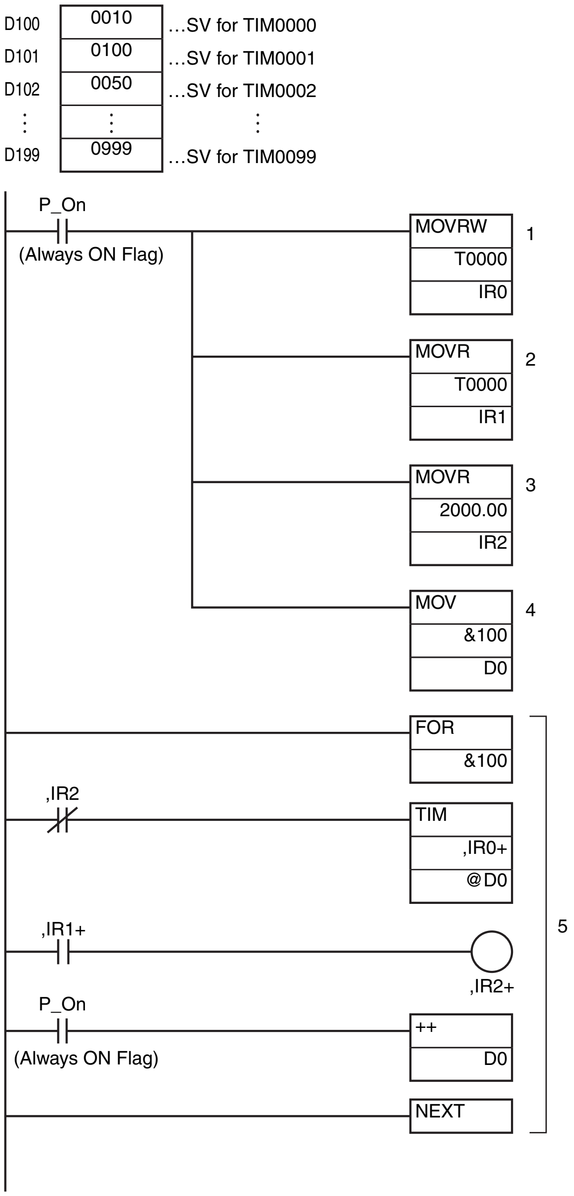
- The internal clock pulse (0.1 s, 0.2 s, 1 s) can be used to easily program a flicker circuit.

For information on clock pulses, refer to the CJ-Series CJ2 CPU Unit Software User's Manual (W473).

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 :



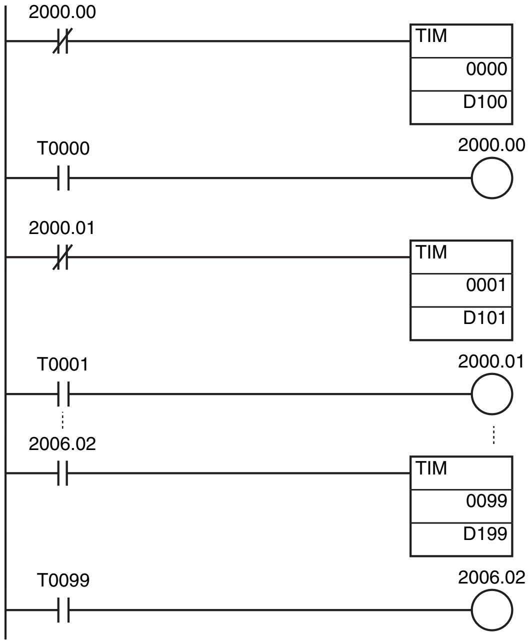
- The following example shows a program section that uses indirect addressing to define and start 100 timers with SVs contained in D100 through D199. IR0 contains the PLC memory address of the timer PV and IR1 contains the PLC memory address of the timer Completion Flag.

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 D100 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 T0000 through T0099.

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



● Timer reset method

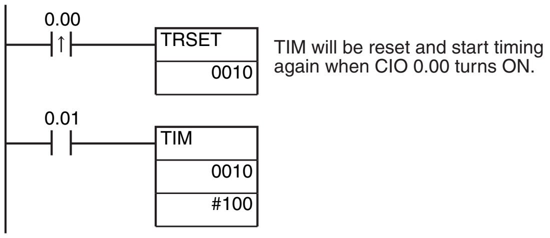
There are three methods for resetting a timer instruction.

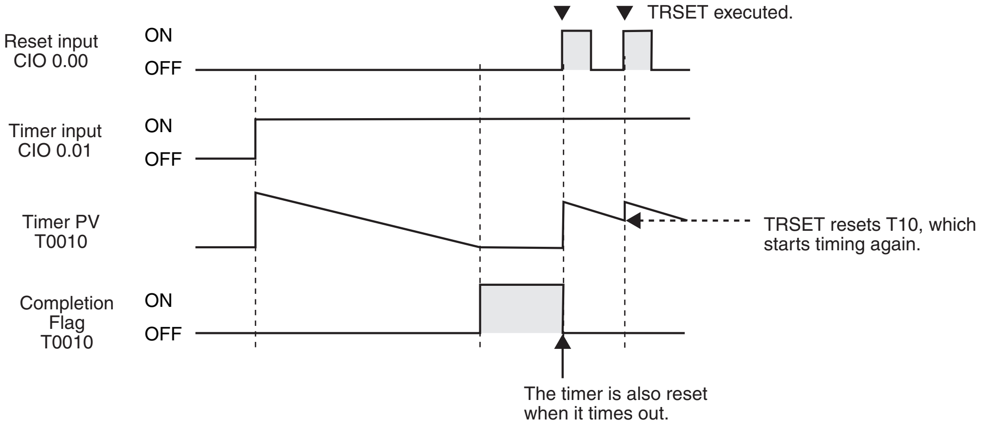
1. Use the TIMER RESET Instruction.

The specified instruction will be reset when the TIMER RESET instruction (TRSET) is executed.

TRSET can be used to reset only one timer at a time, but it is faster than the RESET TIMER/COUNTER instruction (CNR/CNRX).

Use TRSET when a timer is to be reset and then restarted within the same cycle.



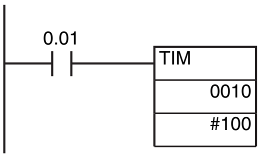


2. Turn OFF the execution condition for the timer instruction.

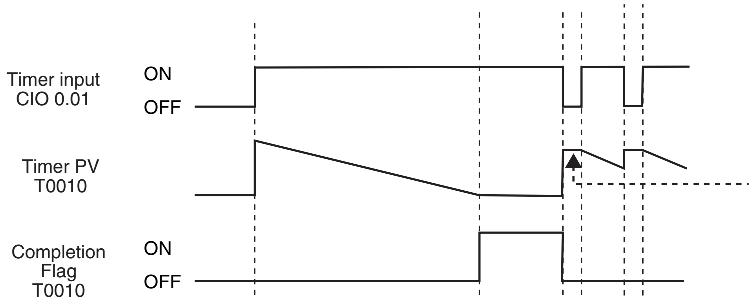
The timer will be reset when its execution condition turns OFF.

The timer will start timing again when its execution condition turns ON.

With this method, a timer cannot be reset and then restarted within the same cycle.



When CIO 0.01 turns ON, the timer starts timing.
When CIO 0.01 turns OFF, the timer is reset.



The timer cannot be reset and then restarted within the same cycle.

3. Use the RESET TIMER/COUNTER instruction.

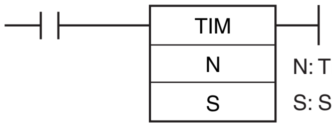
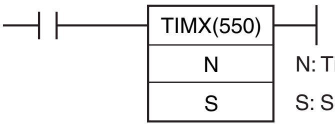
The specified instruction will be reset when the RESET TIMER/COUNTER instruction (CNR/CNRX) is executed.

More than one timer can be reset at the same time for CNR/CNRX, but more time is required than for TRSET.

CNR/CNRX is used in the same way as TRSET.

TIM/TIMX

Instruction	Mnemonic	Variations	Function code	Function
HUNDRED-MS TIMER	TIM/TIMX	---	550	TIM or TIMX(550) operates a decrementing timer with units of 0.1-s.

	TIM	TIMX
Symbol	BCD  N: Timer number S: Set value	Binary  N: Timer number S: Set value

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type		Size
		TIM	TIMX	
N	Timer Number	TIMER	TIMER	1
S	Set Value	WORD	UINT	1

N: Timer Number

The timer number must be between 0000 and 4095 (decimal).

S: Set Value (100-ms units)

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

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

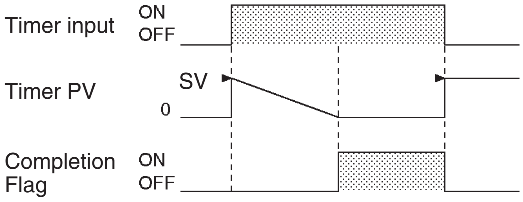
Name	Label	Operation
Error Flag	P_ER	• ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a timer Completion Flag or timer PV. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.
Equals Flag	P_EQ	OFF or unchanged (See note.)
Negative Flag	P_N	OFF or unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.
In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, these Flags are left unchanged.

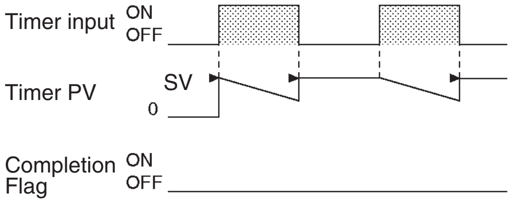
Function

- 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 0.
- 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 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 -0.01 to 0 s.

Note The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from −10 ms to + the cycle time. The timer accuracy for unit version 4.1 of the CJ1-H-R is -0.1 to 0 s.



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.



Hint

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.

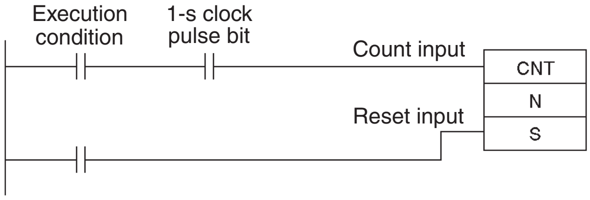
Timers Created with Timer Numbers 0000 to 2047

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 0. The Completion Flag is turned OFF if the PV is not 0.
After execution of all cyclic tasks	The PV is also updated every cycle after executing all cyclic tasks.
80-ms interval refreshing	If the cycle time exceeds 80 ms, the timer’s PV is updated every 80 ms.

Timers Created with Timer Numbers 2048 to 4095

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.
- When the timer set value is #0000, timeup occurs when the instruction is executed.



Precautions

- Timer numbers are shared with other timer 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 2048 to 4095 will not operate properly when the CPU Unit cycle time exceeds 100 ms. Use timer numbers 0 to 2047 when the cycle time is longer than 100 ms.
- The present value of timers programmed with timer numbers 0 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	0	OFF
Power off and reset*2	0	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.

*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 off. However, timing is not continued after power is restored. If the timer input to the instruction is turned ON when power is restored, continuous operation will not be performed.

*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 0000 and 2047 is in a jumped program section (JMP(004), CJMP(510), CJPN(511), JME(005)), the timer's PV will continue timing. (See note.) The jumped TIM/TIMX(550) instruction will not be executed, but the PV will be refreshed each cycle after all tasks have been executed.

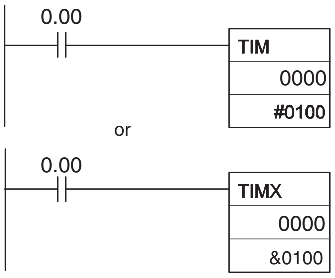
Note With the CS1D CPU Units, the PV will not be refreshed in the above case.

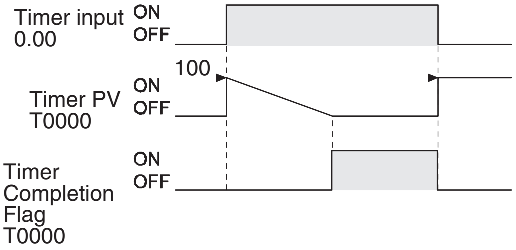
- When a TIM/TIMX(550) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0000. When a TIM/TIMX(550) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.
- The timer's Completion Flag is refreshed only when TIM/TIMX(550) is 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 overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.
- The following restrictions apply if Synchronous Unit Operation is used with a CJ2H-CPU6□-(EIP) CPU Unit or if the internal pulse control period is set to 1 ms for a CJ2M-CPU□□CPU Unit with unit version 2.0 or later.
  - An error of up to one cycle time will occur in the timer PV accuracy.
  - The timers will not operate correctly if the cycle time exceeds 100 ms.
  - If an TIM/TIMX instruction is in a task that is stopped or is not executed because it is jumped by a JMP(004), CJMP(510), or CJPN(511) instruction, the timer will not operate correctly.

Example Programming

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 0.

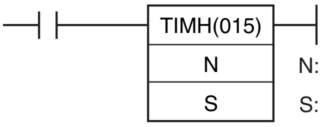
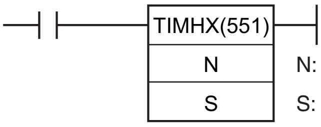
When CIO 0.00 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.





TIMH/TIMHX

Instruction	Mnemonic	Variations	Function code	Function
TEN-MS TIMER	TIMH	---	015	TIMH(015)/TIMHX(551) operates a decrementing timer with units of 10-ms.
	TIMHX	---	551	

	TIMH	TIMHX
Symbol	BCD  N: Timer number S: Set value	Binary  N: Timer number S: Set value

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type		Size
		TIMH	TIMHX	
N	Timer Number	TIMER	TIMER	1
S	Set Value	WORD	UINT	1

N: Timer Number

The timer number must be between 0000 and 4095 (decimal).

S: Set Value

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

TIMHX: &0 to 65535 decimal or #0000 to FFFF hex

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

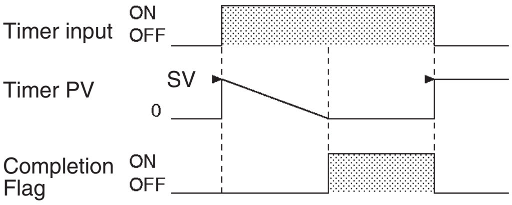
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a timer Completion Flag or timer PV. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.
Equals Flag	P_EQ	Unchanged (See note.)
Negative Flag	P_N	Unchanged (See note.)

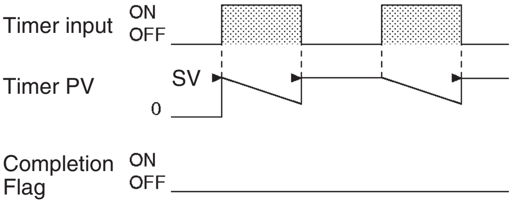
Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

Function

- 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 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 –0.01 to 0 s. The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from –10 ms to + the cycle time.



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.



Hint

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 0000 to 0255

Execution of TIMH(015)/TIMHX(551)	The Completion Flag is turned ON if the PV is 0. The Completion Flag is turned OFF if the PV is not 0.
10-ms interval refreshing	The timer's PV is updated every 10 ms.

Timers Created with Timer Numbers 0256 to 2047

Execution of TIMH(015)/TIMHX(551)	• The PV is updated every time that TIMH(015)/TIMHX(551) 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.
After execution of all cyclic tasks	The PV is also updated every cycle after executing all cyclic tasks
80-ms interval refreshing	If the cycle time exceeds 80 ms, the timer's PV is updated every 80 ms.

Timers Created with Timer Numbers 2048 to 4095

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.

Precautions

- Timer numbers are shared with other timer 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 2048 to 4095 will not operate properly when the CPU Unit cycle time exceeds 100 ms. Use timer numbers 0 to 2047 when the cycle time is longer than 100 ms.
- TIMH(015)/TIMHX(551) timers created with timer numbers 0 to 0255 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 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.
- The Completion Flags for TIMH(015)/TIMHX(551) timers will be updated when the instruction is executed. (This operation differs from that for CV-series and CVM1 PLCs.)
- 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	0	OFF
Power off and reset*2	0	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.

*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 off. However, timing is not continued after power is restored. If the timer input to the instruction is turned ON when power is restored, continuous operation will not be performed.

*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 0000 and 2047 is in a jumped program section (JMP(004), CJMP(510), CJPN(511), JME(005)), the timer's PV will continue timing. (See note.) (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.)

Note With the CS1D CPU Units, the PV will not be refreshed in the above case.

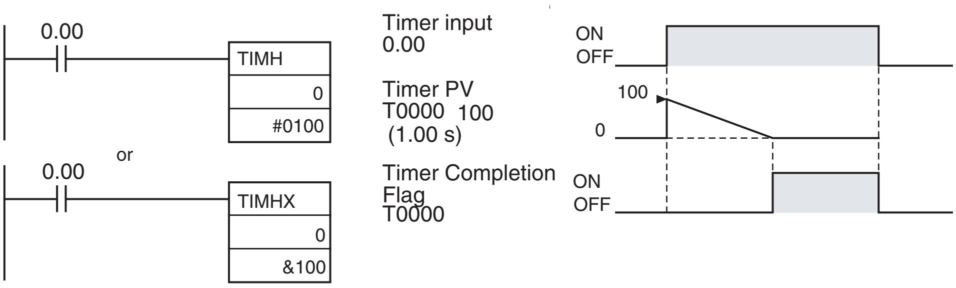
- When TIMH(015)/TIMHX(551) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will be reset to the SV and the Completion Flag will be turned OFF.
- When a TIMH(015)/TIMHX(551) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0000. 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 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 overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.
- The following restrictions apply if Synchronous Unit Operation is used with a CJ2H-CPU6☐-(EIP) CPU Unit or if the internal pulse control period is set to 1 ms for a CJ2M-CPU☐☐ CPU Unit with unit version 2.0 or later.
  - An error of up to one cycle time will occur in the timer PV accuracy.

- The timers will not operate correctly if the cycle time exceeds 100 ms.
- If an TIMH(015)/TIMHX(551) instruction is in a task that is stopped or is not executed because it is jumped by a JMP(004), CJMP(510), or CJPN(511) instruction, the timer will not operate correctly.

Example Programming

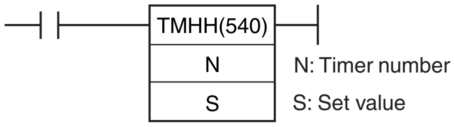
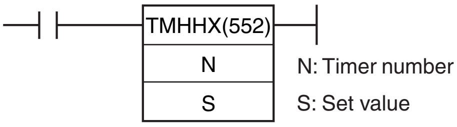
When timer input CIO 0.00 goes from OFF to ON in the following example, the timer PV will begin counting down from the SV (#0064 = 100 = 1.00 s). The Timer Completion Flag, T0000, will be turned ON when the PV reaches 0.

When CIO 0.00 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



TMHH/TMHHX

Instruction	Mnemonic	Variations	Function code	Function
ONE-MS TIMER	TMHH	---	540	TMHH(540)/TMHHX(552) operates a decrementing timer with units of 1-ms.
	TMHHX	---	552	

Symbol	TMHH	TMHHX
	BCD 	Binary 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	Not allowed*1

*1 “OK” on CJ1-H-R/CJ2 CPU Units

Operands

Operand	Description	Data type		Size
		TMHH	TMHHX	
N	Timer Number	TIMER	TIMER	1
S	Set Value	WORD	UINT	1

N: Timer Number

The timer must be between 0000 and 4095 for the CJ1-H-R and CJ2 CPU Units and between 0000 and 0015 decimal for other PLCs.

S: Set Value

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

&0 to 65535 decimal or #0000 to FFFF hex (binary)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a timer Completion Flag or timer PV. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.
Equals Flag	P_EQ	Unchanged (See note.)
Negative Flag	P_N	Unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

Function

- 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).
- 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).

Note The timer accuracy is -0.001 to 0 s. The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from –10 ms to + the cycle time.
The timer accuracy for unit version 4.1 of the CJ1-H-R is –0.01 to 0 s.

Hint

The timer PV and timeup used in TMHH/TMHHX instructions are refreshed at the timing below.

T0000 to T0015[*1]

Refresh timing	Description
1) When each instruction is executed	The timeup flag is ON when the PV is 0 and OFF otherwise.
2) Refresh every 1 ms	Refresh PV every 1 ms

*1 Cannot be used in CJ-H-R CPU Unit Version 4.1.

T0016 to T4095[*2]

Refresh timing	Description
When each instruction is executed	The timeup flag is ON when the PV is 0 and OFF otherwise.

*2 Only with a CJ1-H-R/CJ2 CPU Unit.

Precautions

- Timer numbers are shared with other timer 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 a high-speed timer with a timer number from 0 to 15 will be refreshed even if the task 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]	0	OFF
Power supply interrupted and reset[*2]	0	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.

*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.

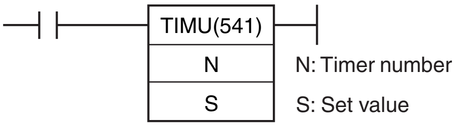
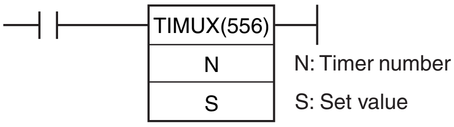
*2 If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted. However, timing is not continued after power is restored. If the timer input to the instruction is turned ON when power is restored, continuous operation will not be performed.

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

- For all CPU Units except CS1D CPU Units, the present value of all operating timers with timer numbers 0 to 15 will be refreshed even if the timer is in a program section that is jumped using JMP(004), CJMP(510), CJPN(511), JME(005). (The jumped timer instruction will not be executed, but the PV will be refreshed every 1 ms.) The present values will not be updated with a CS1D CPU Unit.
- 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 0. 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.
- If online editing is used to overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.
- The following restrictions apply if Synchronous Unit Operation is used with a CJ2H-CPU6□-(EIP) CPU Unit or if the internal pulse control period is set to 1 ms for a CJ2M-CPU□□ CPU Unit with unit version 2.0 or later.
  - An error of up to one cycle time will occur in the timer PV accuracy.
  - The timers will not operate correctly if the cycle time exceeds 100 ms.
  - If an TMHH(540)/TIMHHX(552) instruction is in a task that is stopped or is not executed because it is jumped by a JMP(004), CJMP(510), or CJPN(511) instruction, the timer will not operate correctly.

TIMU/TIMUX

Instruction	Mnemonic	Variations	Function code	Function
TENTH-MS TIMER	TIMU	---	541	TIMU(541)/TIMUX(556) operates a decrementing timer with units of 0.1-ms.
	TIMUX	---	556	

	TIMU	TIMUX
Symbol	BCD 	Binary 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type		Size
		TIMU	TIMUX	
N	Timer Number	TIMER	TIMER	1
S	Set Value	WORD	UINT	1

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).

&0 to 65535 decimal or #0000 to FFFF hex (Binary).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if timer number N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a timer's Completion Flag or PV. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.
Equals Flag	P_EQ	Unchanged
Negative Flag	P_N	Unchanged

Function

- When the timer input is OFF, the timer specified by N is reset, i.e., the timer's Completion Flag is turned OFF.
- When the timer input goes from OFF to ON, TIMU(541)/TIMUX(556) starts decrementing the PV. If the set value is reached while the timer input is ON, the timer's Completion Flag will be turned ON (the timer times out).
- The status of the timer's 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.
- Read this timer's Completion Flag only. The timer's PV is used by the system, so it cannot be read.
- The setting range for the set value (SV) is 0 to 0.9999 s for TIMU(541) and 0 to 6.5535 s for TIMUX(556).
- The timer accuracy is -0.1 to 0 ms.

Hint

- A TIMU(541)/TIMUX(556) instruction's Completion Flag is refreshed as shown in the following table.

Execution of TIMU(541)/TIMUX(556)	The Completion Flag is turned ON if the SV is reached. The Completion Flag is turned OFF if the SV has not been reached.

Precautions

- Timer numbers are shared with other timer 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 timer PV cannot be read.
- The Completion Flag is updated only when TIMU(541)/TIMUX(556) is executed. The Completion Flag can thus be delayed by up to one cycle time from the actual set value.
- The timer will not operate properly when the cycle time exceeds 100 ms.
- 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	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa. (See note 1.)	OFF
Power supply interrupted and reset (See note 2.)	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions3	OFF
Operation in interlocked program section (IL(002)-ILC(003))	OFF
Operation in jumped program section (JMP(004)-JME(005))	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.

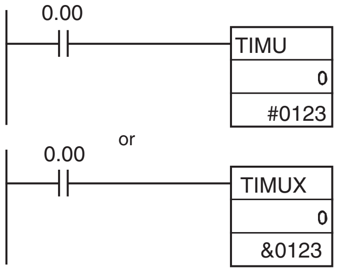
2 If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted. However, timing is not continued after power is restored. If the timer input to the instruction is turned ON when power is restored, continuous operation will not be performed.

- When TIMU(541)/TIMUX(556) 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.
- TIMU(541)/TIMUX(556) timers may not time accurately when used in a program section jumped by the JMP(004), CJMP(510), CJPN(511), and JME(005) instructions.
- When a TIMU(541)/TIMUX(556) timer is forced set, its Completion Flag will be turned ON. When a TIMU(541)/TIMUX(556) timer is forced reset, its Completion Flag will be turned OFF.
- If online editing is used to overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

Example Programming

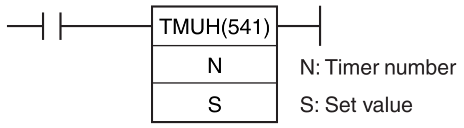
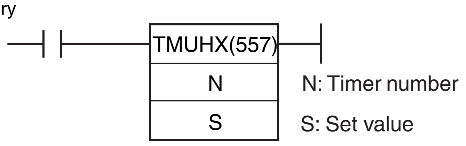
When timer input CIO 0.00 goes from OFF to ON in this example, the timer PV will begin counting down. The Timer Completion Flag, T0000, will be turned ON after 12.3 ms.

When CIO 0.00 goes OFF, the Timer Completion Flag, T0000, will be turned OFF.



TMUH/TMUHX

Instruction	Mnemonic	Variations	Function code	Function
HUNDREDTH-MS TIMER	TMUH	---	544	TMUH(544)/TMUHX(557) operates a decrementing timer with units of 0.01-ms.
	TMUHX	---	557	

<table>
<tr><th rowspan="2">Symbol</th><th>TMUH</th><th>TMUHX</th></tr>
<tr><td>BCD
 </td><td>Binary
 </td></tr>
</table>

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type		Size
		TMUH	TMUHX	
N	Timer Number	TIMER	TIMER	1
S	Set Value	WORD	UINT	1

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).

&0 to 65535 decimal or #0000 to FFFF hex (Binary).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if timer number N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a timer's Completion Flag or PV. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.
Equals Flag	P_EQ	Unchanged
Negative Flag	P_N	Unchanged

Function

- When the timer input is OFF, the timer specified by N is reset, i.e., the timer's Completion Flag is turned OFF.
- When the timer input goes from OFF to ON, TMUH(544)/TMUHX(557) starts decrementing the PV. If the set value is reached while the timer input is ON, the timer's Completion Flag will be turned ON (the timer times out).
- The status of the timer's 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.
- Read this timer's Completion Flag only. The timer's PV is used by the system, so it cannot be read.
- The setting range for the set value (SV) is 0 to 0.09999 s for TMUH(544) and 0 to 0.65535 s for TMUHX(557).
- The timer accuracy is -0.01 to 0 ms.

Hint

A TIMU(541)/TIMUX(556) instruction's Completion Flag is refreshed as shown in the following table.

Execution of TIMU(541)/TIMUX(556)	The Completion Flag is turned ON if the SV is reached. The Completion Flag is turned OFF if the SV has not been reached.

Precautions

- Timer numbers are shared with other timer 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 timer PV cannot be read.
- The Completion Flag is updated only when TIMU(541)/TIMUX(556) is executed. The Completion Flag can thus be delayed by up to one cycle time from the actual set value.
- The timer will not operate properly when the cycle time exceeds 10 ms.
- 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	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa. (See note 1.)	OFF
Power supply interrupted and reset (See note 2.)	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions	OFF
Operation in interlocked program section (IL(002)-ILC(003))	OFF
Operation in jumped program section (JMP(004)-JME(005))	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.

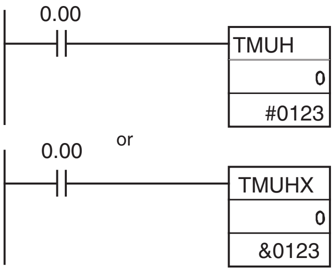
2 If the IOM Hold Bit (A500.12) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted. However, timing is not continued after power is restored. If the timer input to the instruction is turned ON when power is restored, continuous operation will not be performed.

- When TIMU(541)/TIMUX(556) 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.
- TIMUH(544)/TIMUHX(557) timers may not time accurately when used in a program section jumped by the JMP(004), CJMP(510), CJPN(511), and JME(005) instructions.
- When a TIMU(541)/TIMUX(556) timer is forced set, its Completion Flag will be turned ON. When a TIMU(541)/TIMUX(556) timer is forced reset, its Completion Flag will be turned OFF.
- If online editing is used to overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

Example Programming

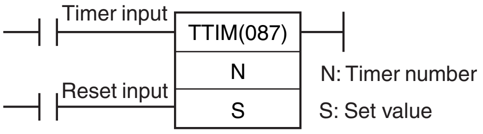
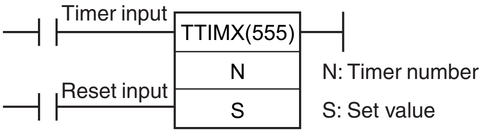
When timer input CIO 0.00 goes from OFF to ON in this example, the timer PV will begin counting down. The Timer Completion Flag, T0000, will be turned ON after 1.23 ms.

When CIO 0.00 goes OFF, the Timer Completion Flag, T0000, will be turned OFF.



TTIM/TTIMX

Instruction	Mnemonic	Variations	Function code	Function
ACCUMULATIVE TIMER	TTIM	---	087	TTIM(087)/TTIMX(555) operates an incrementing timer with units of 0.1-s.
	TTIMX	---	555	

	TTIM	TTIMX
Symbol	BCD 	Binary 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type		Size
		TTIM	TTIMX	
N	Timer Number	TIMER	TIMER	1
S	Set Value	WORD	UINT	1

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).

&0 to 65535 decimal or #0000 to FFFF hex (Binary).

● Operand Specifications

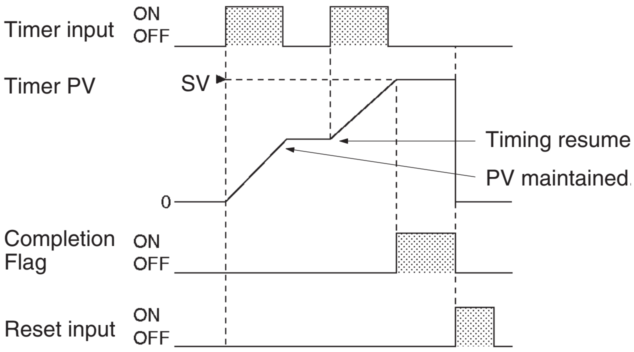
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

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

Function

- 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.
- 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.
  The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from −10 ms to + the cycle time.



Hint

- 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.

Precautions

- Timer numbers are shared with other timer 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 0 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]	0	OFF
Power supply interrupted and reset[*2]	0	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.

*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. However, timing is not continued after power is restored. If the timer input to the instruction is turned ON when power is restored, continuous operation will not be performed.

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

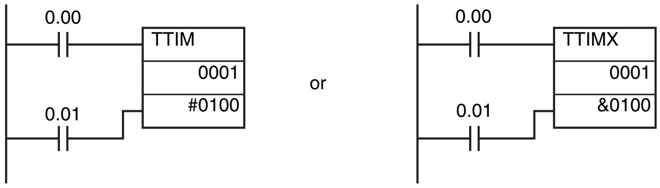
- When TTIM(087)/TTIMX(555) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will retain its previous value (it will not be reset). Be sure to take this fact into account when TTIM(087)/TTIMX(555) is programmed between IL(002) and ILC(003).
- When an 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 0. When a TTIM(087)/TTIMX(555) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to 0. 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.

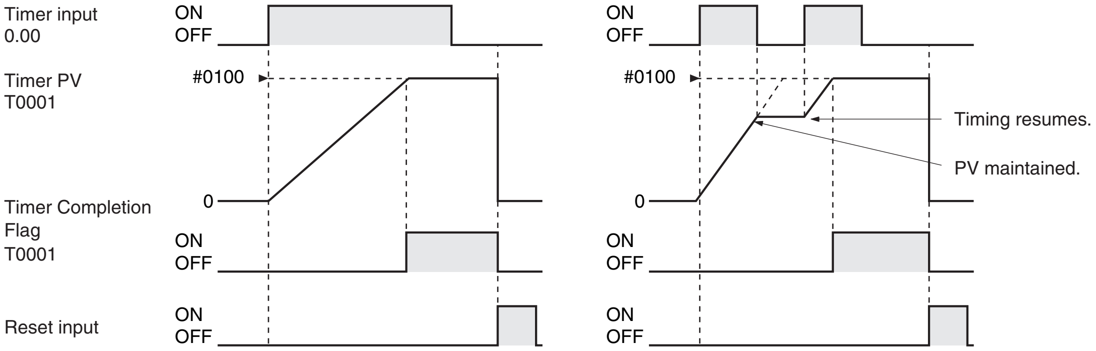
Example Programming

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 0 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.)

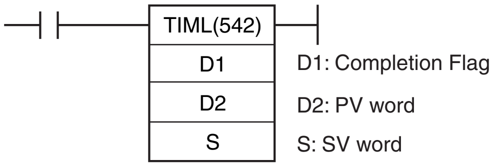
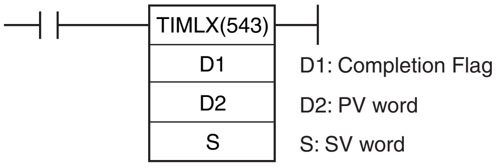
If the timer input is turned OFF before the SV is reached, the timer will stop timing but the PV will be maintained. The timer will resume from its previous PV when the timer input is turned ON again.





TIML/TIMLX

Instruction	Mnemonic	Variations	Function code	Function
LONG TIMER	TIML	---	542	TIML(542)/TIMLX(553) operates a decrementing timer with units of 0.1s.
	TIMLX	---	553	

	TIML	TIMLX
Symbol	BCD 	Binary 

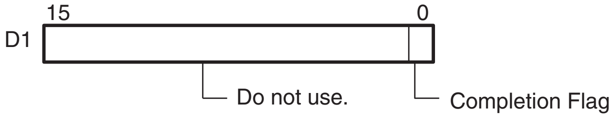
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

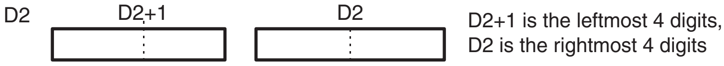
Operands

Operand	Description	Data type		Size
		TIML	TIMLX	
D1	Completion Flag	WORD	UINT	1
D2	PV word	DWORD	UDINT	2
S	SV word	DWORD	UDINT	2

D1: Completion Flag



D2: PV Word



S: SV Word

The PV and SV can range from #00000000 to #99999999 for TIML(542) and &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hexadecimal) for TIMLX(553).

Note S, S+1, D2 and D2+1 must be in the same data area.

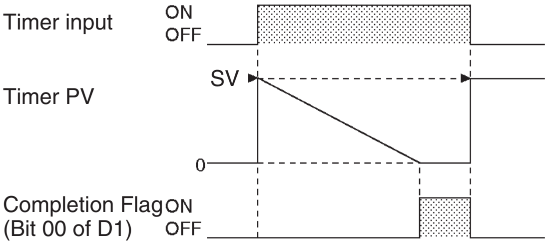
● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D1	OK	OK	OK	OK	---	---	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D2																		
S					OK	OK					OK							

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the PV contained in D2 is not BCD. • ON if the SV contained in S is not BCD, when TIML(542) is used. • OFF in all other cases.

Function



- 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 0.
- 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 timer accuracy is –0.01 to 0 s. The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from –10 ms to + the cycle time.
- TIML(542)/TIMLX(553) can time up to 115 days for TIML(542) and 4,971 days for TIMLX(543).

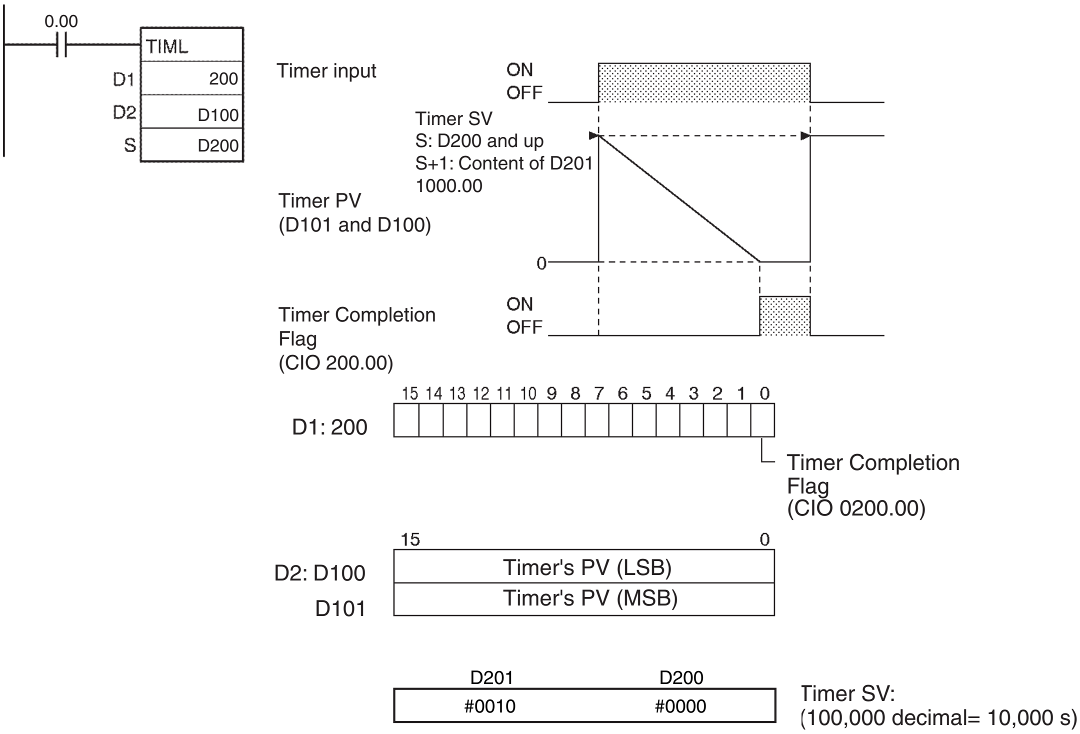
Precautions

- 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 Programming

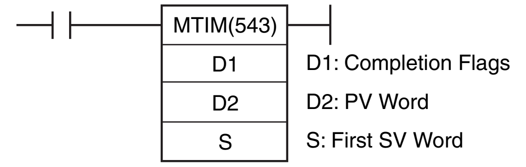
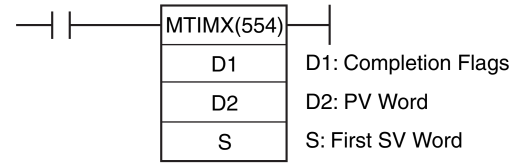
When timer input CIO 0.00 is ON in the following example, the timer PV (in D201 and D200) will be set to the SV (in D101 and D100) and the PV will begin counting down. The timer Completion Flag (CIO 200.00) will be turned ON when the PV reaches 0.

When CIO 0.00 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



MTIM/MTIMX

Instruction	Mnemonic	Variations	Function code	Function
MULTI-OUTPUT TIMER	MTIM	---	543	MTIM(543)/MTIMX(554) operates a 0.1-s incrementing timer with eight independent SVs and Completion Flags.
	MTIMX	---	554	

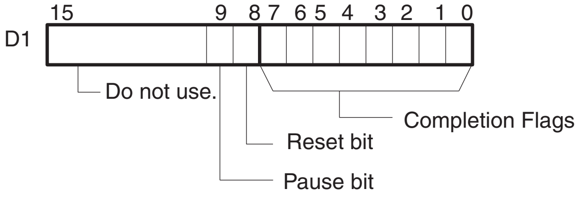
	MTIM	MTIMX
Symbol	BCD 	Binary 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

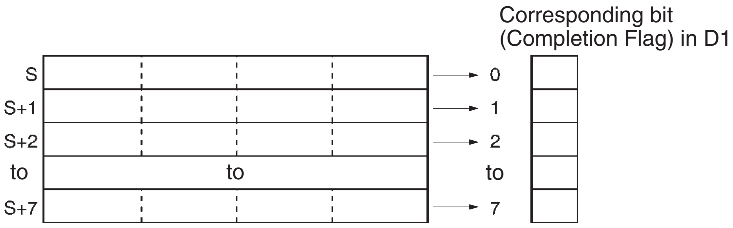
Operand	Description	Data type		Size
		MTIM	MTIMX	
D1	Completion Flags	UINT	UINT	1
D2	PV Word	WORD	UINT	1
S	First SV Word	WORD	WORD	8



Range of D2, S to S+7

- In BCD:
  #0000 to #9999
- In Binary:
  &0 to &65535 (decimal) or
  #0000 to #FFFF (hex)

Note S through S+7 must be in the same data area.



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D2												OK						
S												---						

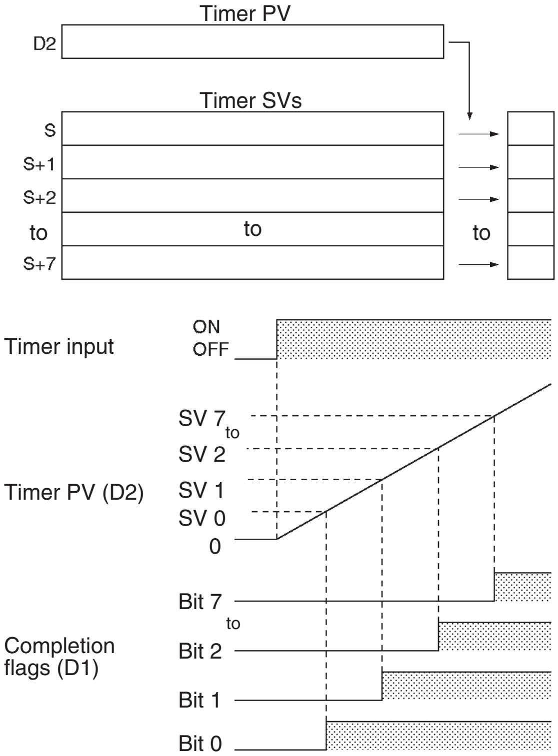
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the PV contained in D2 is not BCD, when MTIM(543) is used. • OFF in all other cases.

Function

- 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 0 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 0 and all of the Completion Flags will be turned OFF.
- 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 –0.01 to 0 s.

Note The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from –10 ms to + the cycle time.



- 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 0 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

Hint

- 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.

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 0 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 0. MTIM(543)/MTIMX(554) will ignore the SV that is set to 0 and all of the remaining SVs.

SCH	2CH	1	0	2	9
	3CH	2	5	0	6
	4CH	6	0	4	0
	5CH	0	0	0	0
	to	to			
S+7CH	9CH				

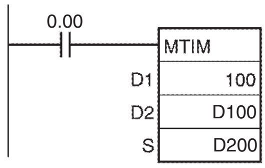
These SVs are ignored.

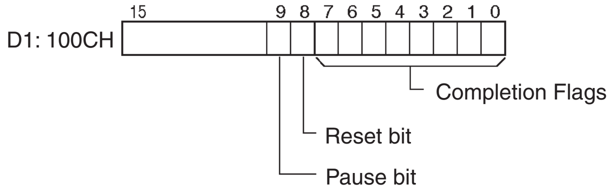
- The timer's PV is refreshed only when MTIM(543)/MTIMX(554) is executed, so the timer will not operate properly when the cycle time exceeds 100 ms because the timer increments in 100-ms units. To ensure precise timing and prevent problems caused by long cycle times, input the same MTIM(543)/MTIMX(554) 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.

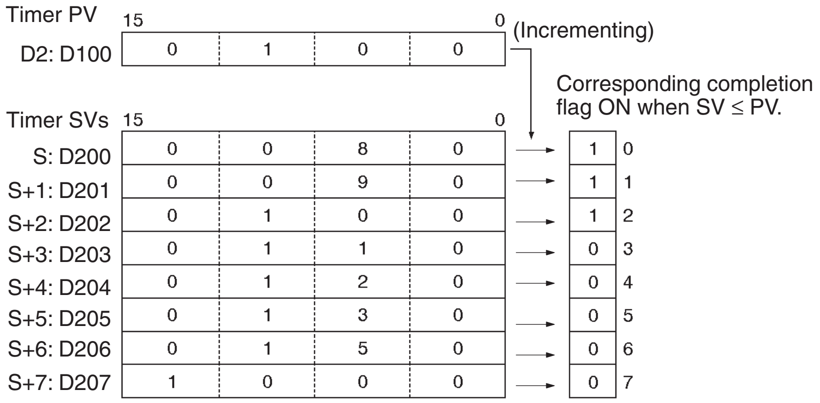
Example Programming

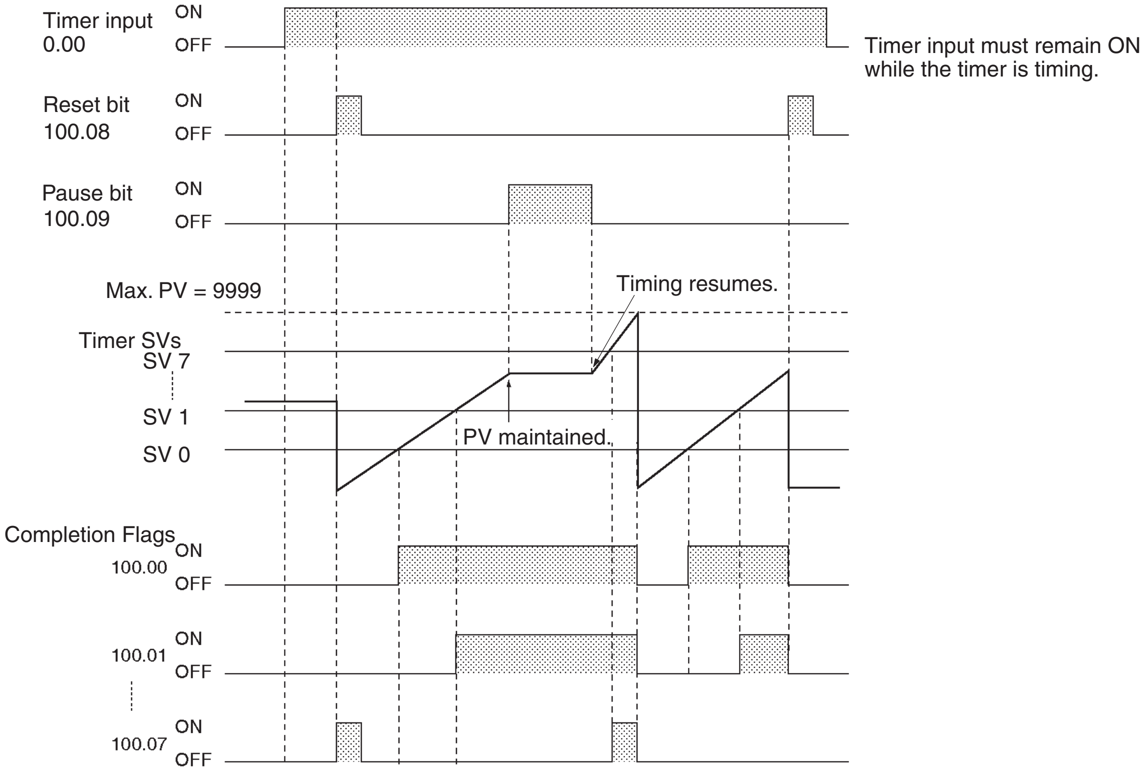
When CIO 0.00 is ON and the pause bit (CIO 010009) is OFF in the following example, the timer will start operating when the reset bit (CIO 010009) is turned from ON to OFF. The timer's PV will begin timing up from 0.

The eight SVs in D200 through D207 are compared to the PV and the corresponding Completion Flags (CIO 010000 through CIO 010007) are turned on when the SV ≤ PV.


0.00
MTIM
D1
100
D2
D100
S
D200

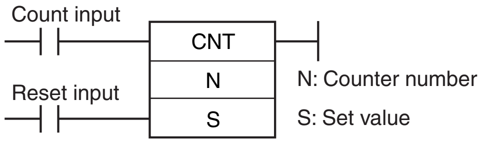
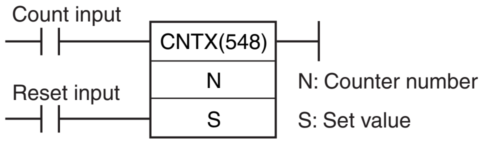

15
9 8 7 6 5 4 3 2 1 0
D1: 100CH
Completion Flags
Reset bit
Pause bit


Timer PV
D2: D100
15
0
0 1 0 0
(Incrementing)
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


Timer input
0.00
ON
OFF
Timer input must remain ON while the timer is timing.
Reset bit
100.08
ON
OFF
Pause bit
100.09
ON
OFF
Max. PV = 9999
Timing resumes.
Timer SVs
SV 7
SV 1
SV 0
PV maintained.
Completion Flags
100.00
ON
OFF
100.01
ON
OFF
100.07
ON
OFF

CNT/CNTX

Instruction	Mnemonic	Variations	Function code	Function
COUNTER	CNT/CNTX	---	546	CNT/CNTX(546) operates a decrementing counter.

	CNT	CNTX
Symbol	BCD 	Binary 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size
		CNT	CNTX	
N	Counter Number	COUNTER	COUNTER	1
S	Set Value	WORD	UINT	1

N: Counter Number

The counter number must be between 0000 and 4095 (decimal).

S: Set Value

BCD: #0000 to #9999
Binary: &0 to &65535 (decimal) or #0000 to #FFFF (hex)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	OK	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK						

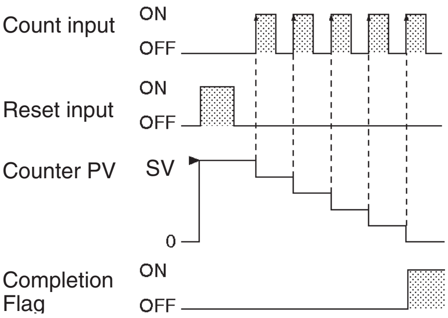
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a counter Completion Flag or counter PV. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.
Equals Flag	P_EQ	Unchanged (See note.)
Negative Flag	P_N	Unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

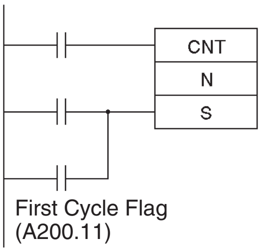
Function

- 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.)
- The setting range 0 to 9,999 for CNT and 0 to 65,535 for CNTX(546).



Hint

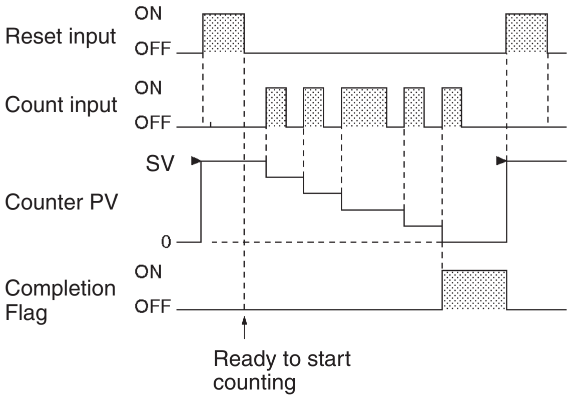
- Counter PVs are retained even through a power interruption. If you want to restart counting from the SV instead of resuming the count from the retained PV, add the First Cycle Flag (A200.11) as a reset input to the counter.



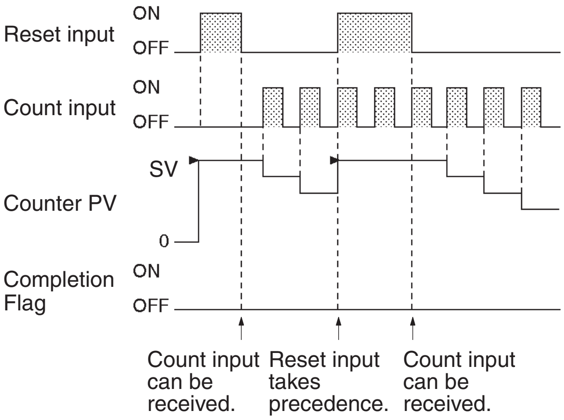
Precautions

- 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.



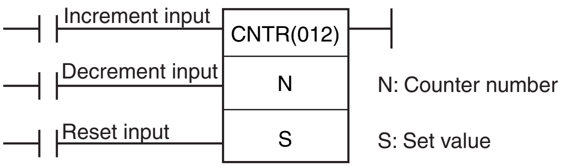
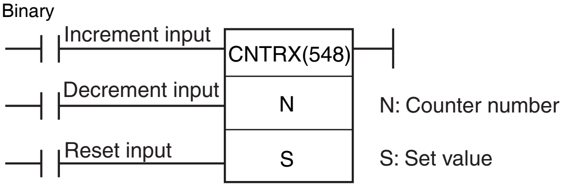
- The reset input will take precedence and the counter will be reset if the reset input and count input are both ON at the same time. (The PV will be reset to the SV and the Completion Flag will be turned OFF.)



- 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.

CNTR/CNTRX

Instruction	Mnemonic	Variations	Function code	Function
REVERSIBLE COUNTER	CNTR	---	012	---
	CNTRX	---	548	

	CNTR	CNTRX
Symbol	BCD  	Binary 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size
		CNTR	CNTRX	
N	Counter Number	COUNTER	COUNTER	1
S	Set Value	WORD	UINT	1

N: Counter Number

The counter number must be between 0000 and 4095 (decimal).

S: Set Value

BCD:#0000 to #9999
Binary: &0 to &65535 (decimal) or #0000 to #FFFF (hex)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	OK	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK						

Flags

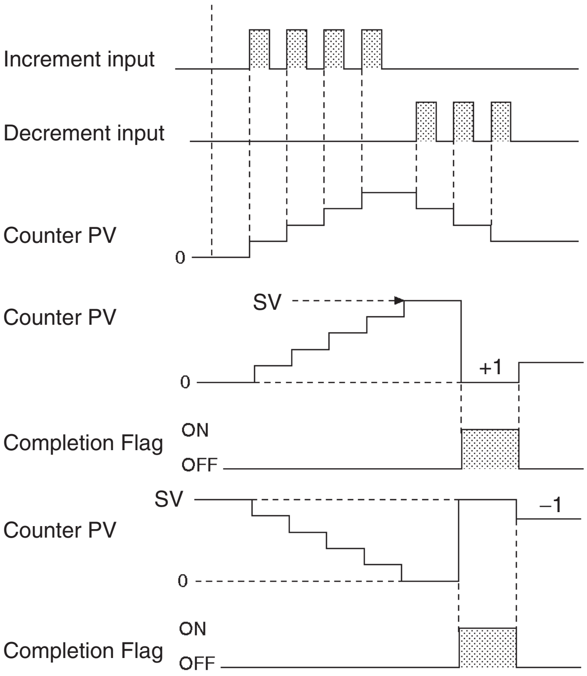
Name	Label	Operation
Error Flag	P_ER	• ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a counter. • ON if in BCD mode and S does not contain BCD data. • OFF in all other cases.

Function

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.



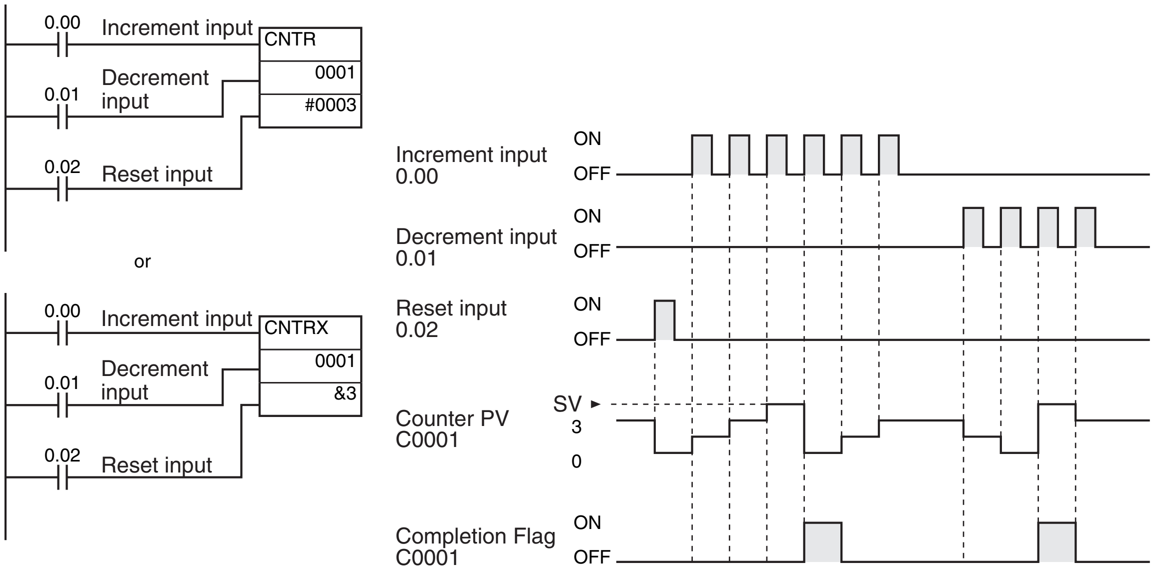
Precautions

- 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).

Example Programming

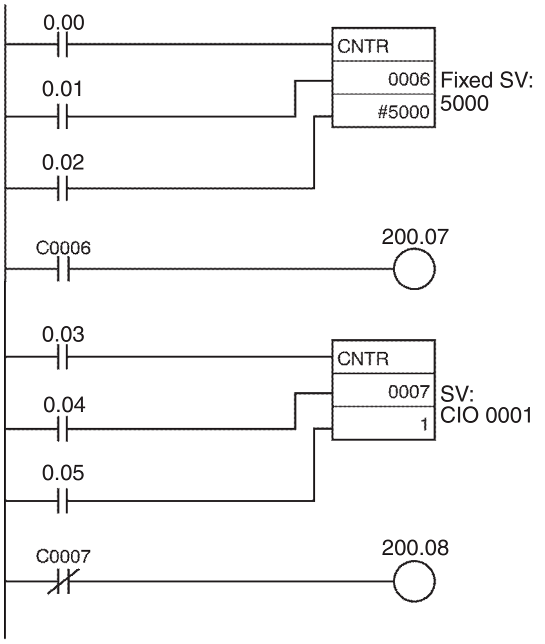
The counter PV is reset to 0 by turning the reset input (CIO 0.02) ON and OFF. The PV is incremented by 1 each time that the increment input (CIO 0.00) goes from OFF to ON. When the PV is incremented from the SV (3), it is automatically reset to 0 and the Completion Flag is turned ON.

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.

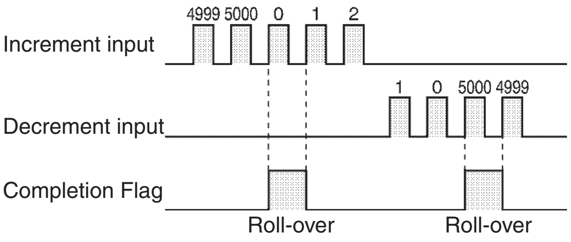


The add and subtract count inputs increase/decrease the count once when the signal rises (OFF to ON). When both inputs turn ON at the same time, neither increases/decreases the count. When the reset input turns ON, the PV changes to 0 and count input is not accepted.

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

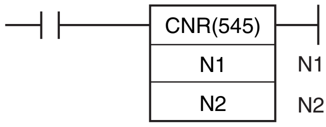
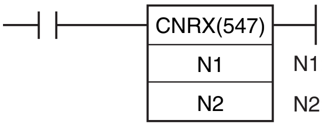


Instruction	Operands
LD	0.00
LD	0.01
LD	0.02
CNTR (012)	0006
	#5000
LD	200.07
OUT	0.03
LD	0.04
LD	0.05
LD	0007
CNTR (012)	1
LD NOT	C0007
OUT	200.08



CNR/CNRX

Instruction	Mnemonic	Variations	Function code	Function
RESET TIMER/COUNTER	CNR	@CNR	545	Resets the timers or counters within the specified range of timer or counter numbers.
	CNRX	@CNRX	547	

	CNR	CNRX
Symbol	BCD  CNR(545) N1 N2 N1: First number in range N2: Last number in range	Binary  CNRX(547) N1 N2 N1: First number in range N2: Last number in range

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size
		CNT	CNTX	
N1	First number in range	TIMER/COUNTER*1		Variable
N2	Last number in range	TIMER/COUNTER*1		Variable

N1: First Number in Range

N1 must be a timer number between T0000 and T4095 or a counter number between C0000 and C4095.

N2: Last Number in Range

N2 must be a timer number between T0000 and T4095 or a counter number between C0000 and C4095

Note N1 and N2 must be in the same data area, i.e., N1 and N2 must be timer numbers or counter numbers.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N1	---	---	---	---	OK	OK	---	---	---	---	---	---	---	OK	---	---	---	---
N2																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N1 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 N2 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 N1 and N2 are not in the same data area. • OFF in all other cases.

Function

CNR(545)/CNRX(547) resets the Completion Flags of all timers or counters from N1 to N2. 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.)

Precautions

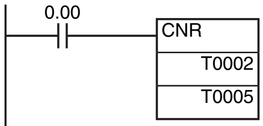
• The timer/counter that is reset is as follows.

	Instructions reset	Operation of CNR(545)
BCD	TIM: HUNDRED-MS TIMER TIMH(015): TEN-MS TIMER TMHH(540): ONE-MS TIMER TTIM(087): ACCUMULATIVE TIMER TIMW(813): HUNDRED-MS TIMER WAIT TMHW(815): TEN-MS TIMER WAIT CNT: COUNTER CNTR(012): REVERSIBLE COUNTER CNTW(814): COUNTER WAIT	The PV is set to its maximum value (9,999 BCD) and the Completion Flag is turned OFF.
	TIMU(541): TENTH-MS TIMER TMUH(544): HUNDREDTH-MS TIMER	The Completion Flag is turned OFF. (The PV cannot be read.)

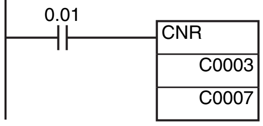
	Instructions reset	Operation of CNRX(547)
Binary	TIMX(550): HUNDRED-MS TIMER TIMHX(551): TEN-MS TIMER TMHHX(552): ONE-MS TIMER TTIMX(555): ACCUMULATIVE TIMER TIMWX(816): HUNDRED-MS TIMER WAIT TMHWX(817): TEN-MS TIMER WAIT CNTX(546): COUNTER CNTRX(548): REVERSIBLE COUNTER CNTWX(818): COUNTER WAIT	The PV is set to its maximum value (FFFF hex) and the Completion Flag is turned OFF.
	TIMUX(556): TENTH-MS TIMER TMUHX(557): HUNDREDTH-MS TIMER	The Completion Flag is turned OFF. (The PV cannot be read.)

- The CNR(545)/CNRX(547) instructions do not reset TIML(542), TIMLX(553), MTIM(543), and MTIMX(554), because these timers do not use timer numbers.
- The CNR(545)/CNRX(547) instructions do not reset the timer/counter instructions themselves, they reset 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 (9999 for BCD and FFFF for binary).
- When N1 and N2 are specified with N1>N2, only the Completion Flag for the timer/counter number will be reset.

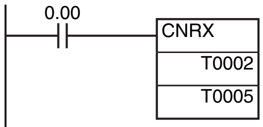
Example Programming



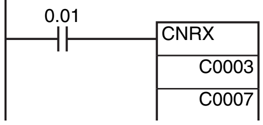
When CIO 0.00 is ON in the following example, the Completion Flags for timers T0002 to T0005 are turned OFF and the timers' PVs are set to the maximum value (9999 for BCD).



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 (9999 for BCD).



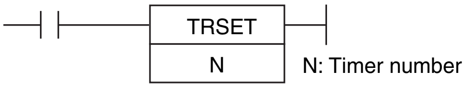
When CIO 0.00 is ON in the following example, the Completion Flags for timers T0002 to T0005 are turned OFF and the timers' PVs are set to the maximum value (FFFF for binary).



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 (FFFF for binary).

TRSET

Instruction	Mnemonic	Variations	Function code	Function
TIMER RESET	TRSET	@TRSET	549	Resets the specified timer.

Symbol	TRSET
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Timer number	TIMER	1

● Operand Specifications

Area	Word addresses							Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the address of a timer Completion Flag or timer PV. • OFF in all other cases.
Equals Flag	P_EQ	Unchanged.
Negative Flag	P_N	Unchanged.

Function

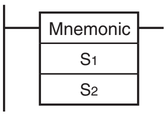
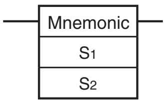
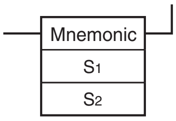
Resets the specified timer.

Comparison Instructions

=, <>, <, <=, >, >=

Instruction	Mnemonic	Variations	Function code	Function
Input Comparison Instructions	=, <>, <, <=, >, >=	---	300 to 328	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.

Note Input comparison instructions are available to compare signed or unsigned data of one-word or double length data. Refer to 3-15-24 Single-precision Floating-point Comparison Instructions for details on single-precision floating-point input comparison instructions and 3-16-21 Double-precision Floating-point Input Comparison Instructions for details on double-precision floating-point input comparison instructions.

	=, <>, <, <=, >, >=		
Symbol	LD connection  S1: Comparison data 1 S2: Comparison data 2	AND connection  S1: Comparison data 1 S2: Comparison data 2	OR connection  S1: Comparison data S2: Comparison data

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type				Size	
		Unsigned	Unsigned double length	Signed	Signed double length	One-word	Double length
S1	Comparison data 1	UINT	UDINT	INT	DINT	1	2
S2	Comparison data 2	UINT	UDINT	INT	DINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
One-word Data	S1 S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
Double-length Data	S1 S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	OFF or unchanged (See note.)
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.

Name	Label	Operation
Equal Flag	=	• ON if $S_1 = S_2$ with one-word data. • ON if $S_1+1, S_1 = S_2+1, S_2$ with double-length data. • OFF in all other cases.
Not Equal Flag	< >	• ON if $S_1 \neq S_2$ with one-word data. • ON if $S_1+1, S_1 \neq S_2+1, S_2$ with double-length data. • OFF in all other cases.
Less Than Flag	<	• ON if $S_1 < S_2$ with one-word data. • ON if $S_1+1, S_1 < S_2+1, S_2$ with double-length data. • OFF in all other cases.
Less Than or Equal Flag	< =	• ON if $S_1 \leq S_2$ with one-word data. • ON if $S_1+1, S_1 \leq S_2+1, S_2$ with double-length data. • OFF in all other cases.
Negative Flag	N	OFF or unchanged (See note.)

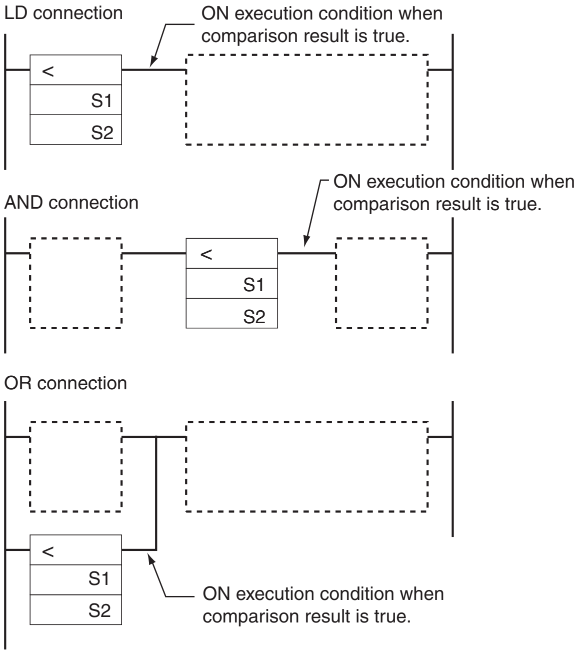
Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.
In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, these Flags are left unchanged.

Function

The input comparison instruction compares S1 and S2 as signed or unsigned values and creates an ON execution condition when the comparison condition is true.

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

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

Summary of Input Comparison Instructions

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

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

Code	Mnemonic	Name	Function
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	

Hint

- 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.

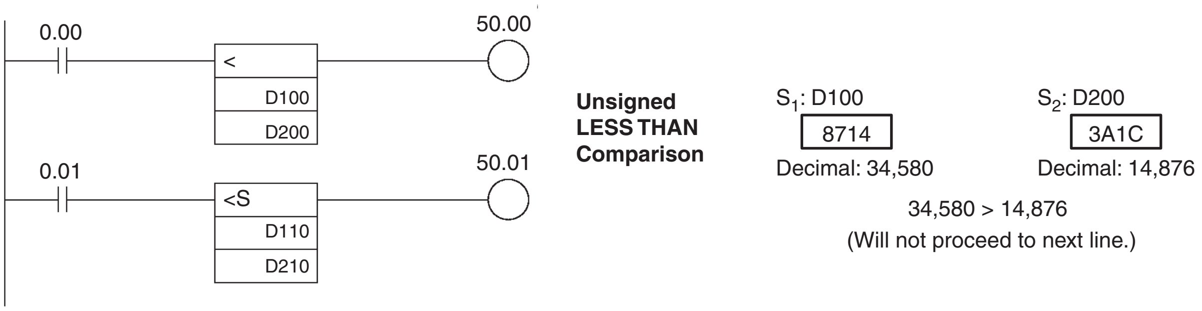
Precautions

- 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 Programming

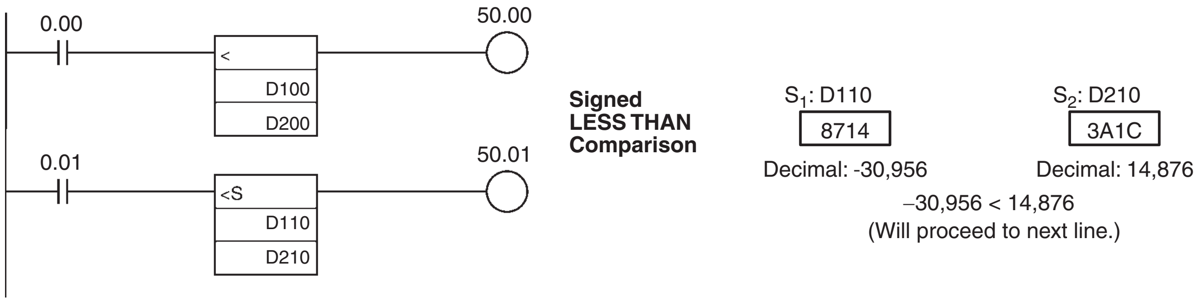
AND LESS THAN: AND<(310)

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 50.00 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.



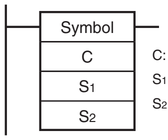
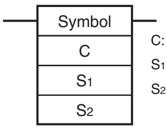
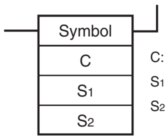
AND SIGNED LESS THAN: AND<S(312)

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 50.01 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.



=DT, <>DT, <DT, <=DT, >DT, >=DT

Instruction	Mnemonic	Variations	Function code	Function
Time Comparison Instructions	=DT <>DT <DT <=DT >DT >=DT	---	341 342 343 344 345 346	Time comparison instructions compare two BCD time values and create an ON execution condition when the comparison condition is true.

	=DT, <>DT, <DT, <=DT, >DT, >=DT		
	LD	AND	OR
Symbol	 C: Control word S_1: First word of present time S_2: First word of comparison time	 C: Control word S_1: First word of present time S_2: First word of comparison time	 C: Control word S_1: First word of present time S_2: First word of comparison time

Applicable Program Areas

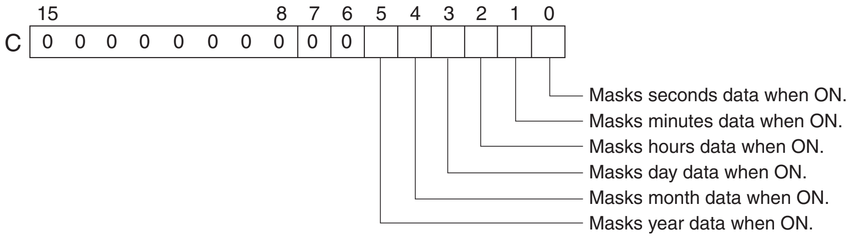
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	WORD	1
S1	First word of present time	WORD	3
S2	First word of comparison time	WORD	3

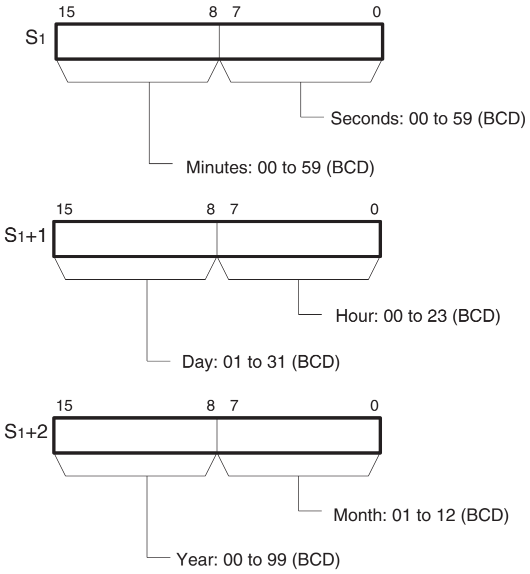
C: Control Word

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

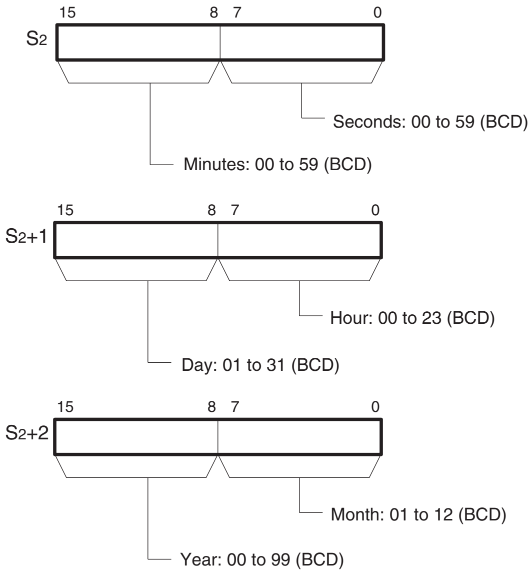
S_1 through S_1+2 contain the present time data. S_1 through S_1+2 must be in the same data area.



Note When using the CPU Unit's internal clock data for the comparison, set S1 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.



Note The year value indicates the last two digits of the year. Values 00 to 97 are interpreted as 2000 to 2097. Values 98 and 99 are interpreted as 1998 and 1999.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
C	OK	OK	OK	OK	OK	OK	---	---	---	---	OK	---	---	OK	---	---	---	---
S1, S2							OK	OK	OK	OK	---							

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if all 6 of the mask bits (C bits 00 to 05) are ON. • OFF in all other cases.
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.

Function

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 (=, <>, <, <=, >, >=).

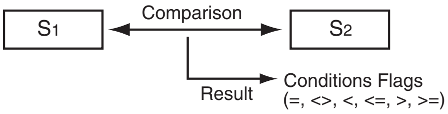
The time comparison instructions are treated just like the LD, AND, and OR instructions to control the execution of subsequent instructions.

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.

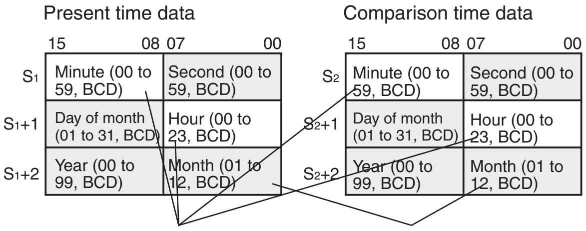
Result	Flag status					
	=	<>	<	<=	>	>=
$S_1 = S_2$	ON	OFF	OFF	ON	OFF	ON
$S_1 > S_2$	OFF	ON	OFF	OFF	ON	ON
$S_1 < S_2$	OFF	ON	ON	ON	OFF	OFF



● 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.



Compares only hours and minutes data.

Year, month, day, and seconds data is masked.

Hint

- 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)

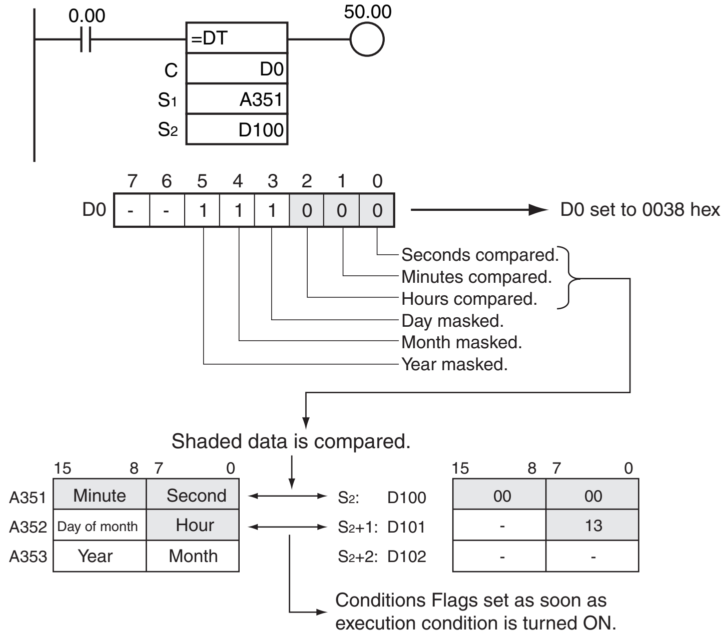
- The Calendar/Clock Area can be set with a Programming Device (including a Programming Console), DATE(735) instruction, or “CLOCK WRITE” FINS command (0702 hex).

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.

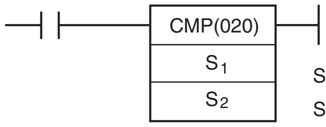
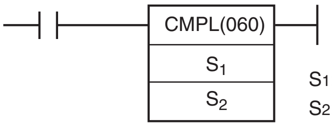
Example Programming

When CIO 0.00 is ON and the time is 13:00:00, CIO 50.00 is turned ON. The contents of A351 to A352 (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.



CMP/CMPL

Instruction	Mnemonic	Variations	Function code	Function
COMPARE	CMP	!CMP	020	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.
DOUBLE COMPARE	CMPL	---	060	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.

	CMP	CMPL
Symbol	 S1: Comparison data 1 S2: Comparison data 2	 S1: Comparison data 1 S2: Comparison data 2

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		CMP	CMPL	CMP	CMPL
S1	CMP: Comparison data 1 CMPL: Comparison data 1, rightmost word address	UINT	UDINT	1	2
S2	CMP: Comparison data 2 CMPL: Comparison data 2, rightmost word address	UINT	UDINT	1	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
CMP S1, S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
CMPL S1, S2												---	OK					

Flags

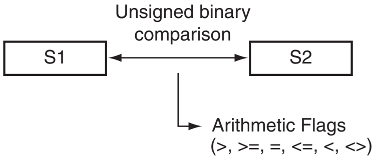
Name	CX-Programmer label	Operation	
		CMP	CMPL
Error Flag	ER	Unchanged (See note.)	Unchanged (See note.)
Greater Than Flag	>	• ON if $S_1 > S_2$. • OFF in all other cases.	• ON if $S_1+1, S_1 > S_2+1, S_2$. • OFF in all other cases.
Greater Than or Equal Flag	> =	• ON if $S_1 \geq S_2$. • OFF in all other cases.	• ON if $S_1+1, S_1 \geq S_2+1, S_2$. • OFF in all other cases.
Equal Flag	=	• ON if $S_1 = S_2$. • OFF in all other cases.	• ON if $S_1+1, S_1 = S_2+1, S_2$. • OFF in all other cases.
Not Equal Flag	=	• ON if $S_1 \neq S_2$. • OFF in all other cases.	• ON if $S_1+1, S_1 \neq S_2+1, S_2$. • OFF in all other cases.
Less Than Flag	<	• ON if $S_1 < S_2$. • OFF in all other cases.	• ON if $S_1+1, S_1 < S_2+1, S_2$. • OFF in all other cases.
Less Than or Equal Flag	< =	• ON if $S_1 \leq S_2$. • OFF in all other cases.	• ON if $S_1+1, S_1 \leq S_2+1, S_2$. • OFF in all other cases.
Negative Flag	N	Unchanged (See note.)	Unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

● The following table shows the status of the Arithmetic Flags after execution of CMP(020).

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

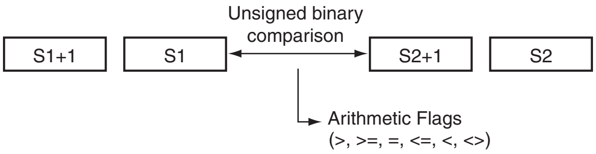
*A status of "---" indicates that the Flag may be ON or OFF.



● The following table shows the status of the Arithmetic Flags after execution of CMPL(060).

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

*A status of "---" indicates that the Flag may be ON or OFF.



Function

● CMP

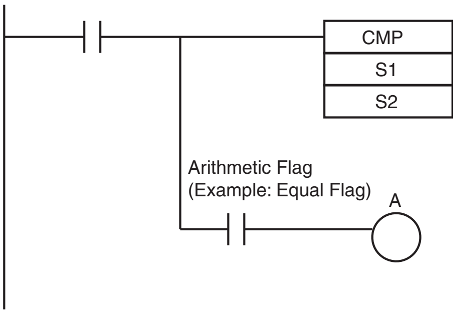
CMP(020) compares the unsigned binary data in S_1 and S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.

● CMPL

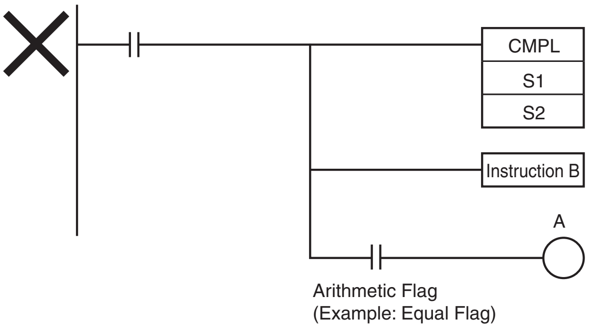
CMPL(060) compares the unsigned binary data in S_1 +1, S_1 and S_2+1, S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.

Precautions

- Using CMP(020)Results in the Program
  When CMP(020)/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 CMP(020)/CMPL(060), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when $S_1 = S_2$ or $S_1 + 1$, $S_1 = S_2 + 1$, S_2.



- Using CMP(020) Results in the Program
  Do not program another instruction between CMP(020)/CMPL(060) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag.

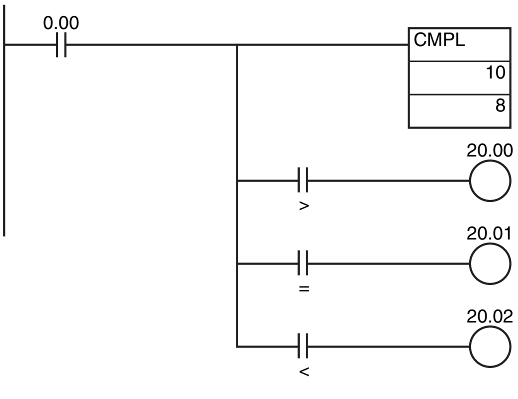


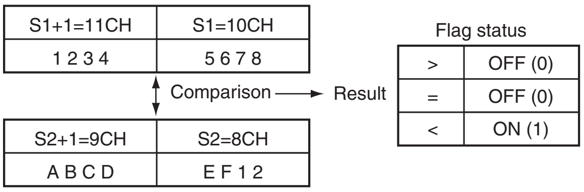
In this case, the results of instruction B might change the results of CMP(020).

- 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. (Immediate refreshing cannot be performed on inputs allocated to Group-2 High-density I/O Units or Units mounted to Slave Racks.)

Example Programming

- When CIO 0.00 is ON in the following example, the eight-digit unsigned binary data in CIO 0011 and CIO 0010 is compared to the eight-digit unsigned binary data in CIO 0009 and CIO 0008 and the result is output to the Arithmetic Flags. The results recorded in the Greater Than, Equals, and Less Than Flags are immediately saved to CIO 20.00 (Greater Than), CIO 20.01 (Equals), and CIO 20.02 (Less Than).





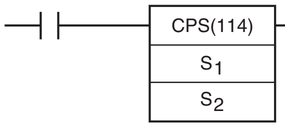
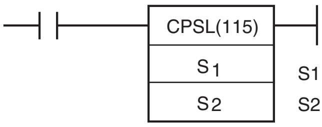
S1+1=11CH	S1=10CH
1 2 3 4	5 6 7 8

S2+1=9CH	S2=8CH
A B C D	E F 1 2

>	OFF (0)
=	OFF (0)
<	ON (1)

CPS/CPSL

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY COMPARE	CPS	!CPS	114	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.
DOUBLE SIGNED BINARY COMPARE	CPSL	---	115	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.

	CPS	CPSL
Symbol	 S1: Comparison data 1 S2: Comparison data 2	 S1: Comparison data 1 S2: Comparison data 2

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		CPS	CPSL	CPS	CPSL
S1	CMP: Comparison data 1 CMPL: Comparison data 1, rightmost word address	INT	DINT	1	2
S2	CMP: Comparison data 2 CMPL: Comparison data 2, rightmost word address	INT	DINT	1	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
CPS S1, S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
CPSL S1, S2												---						

Flags

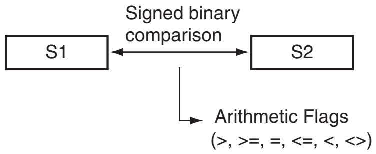
Name	Label	Operation	
		CPS	CPSL
Error Flag	ER	Unchanged (See note.)	OFF or unchanged (See note.)
Greater Than Flag	>	• ON if $S_1 > S_2$. • OFF in all other cases.	• ON if S_1 +1, $S_1 > S_2$+1, S_2. • OFF in all other cases.
Greater Than or Equal Flag	> =	• ON if $S_1 \geq S_2$. • OFF in all other cases.	• ON if S_1 +1, $S_1 \geq S_2$+1, S_2. • OFF in all other cases.
Equal Flag	=	• ON if $S_1 = S_2$. • OFF in all other cases.	• ON if S_1 +1, $S_1 = S_2$+1, S_2. • OFF in all other cases.
Not Equal Flag	=	• ON if $S_1 \neq S_2$. • OFF in all other cases.	• ON if S_1 +1, $S_1 \neq S_2$+1, S_2. • OFF in all other cases.
Less Than Flag	<	• ON if $S_1 < S_2$. • OFF in all other cases.	• ON if S_1 +1, $S_1 < S_2$+1, S_2. • OFF in all other cases.
Less Than or Equal Flag	< =	• ON if $S_1 \leq S_2$. • OFF in all other cases.	• ON if S_1 +1, $S_1 \leq S_2$+1, S_2. • OFF in all other cases.
Negative Flag	N	Unchanged (See note.)	OFF or unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

- **The following table shows the status of the Arithmetic Flags after execution of CPS(114).**

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

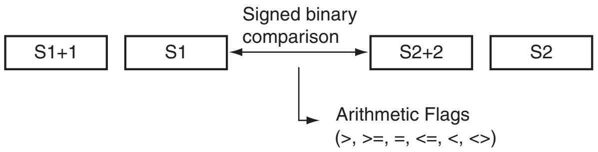
*A status of "---" indicates that the Flag may be ON or OFF.



- **The following table shows the status of the Arithmetic Flags after execution of CPSL(115).**

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

*A status of "---" indicates that the Flag may be ON or OFF.



Function

CPS

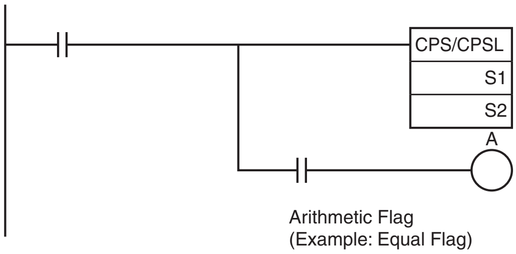
CPS(114) compares the signed binary data in S_1 and S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.

CPSL

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.

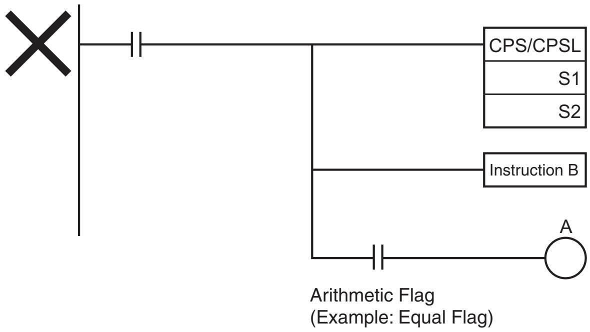
Precautions

- When CPS(114)/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 CPS(114)/CPSL(115), as shown in the following diagram.



In this case, the Equals Flag and output A will be turned ON when $S_1 = S_2$ or $S_1 +1$, $S_1 = S_2+1$, S_2.

- Do not program another instruction between CPS(114)/CPSL(115) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag.



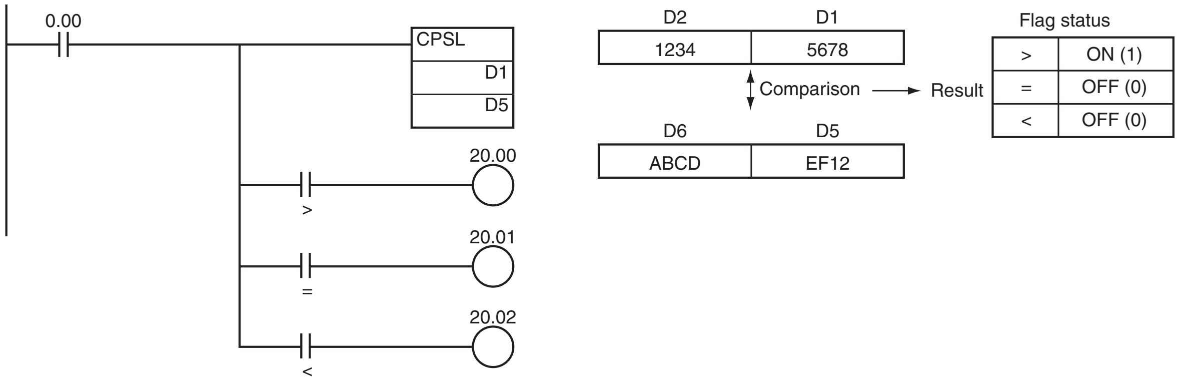
In this case, the results of instruction B might change the results of CPS(114)/CPSL(115).

- The immediate-refreshing variation (!CPS(114)/!CPSL(115)) can be used with words allocated to external inputs specified in S_1 and/or S_2. When !CPS(114)/!CPSL(115) 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. (Immediate refreshing cannot be performed on inputs allocated to Group-2 High-density I/O Units or Units mounted to Slave Racks.)

Example Programming

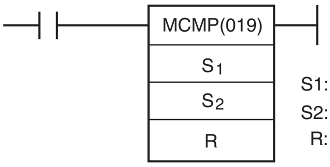
When CIO 0.00 is ON in the following example, the eight-digit signed binary data in D2 and D1 is compared to the eight-digit signed binary data in D6 and D5 and the result is output to the Arithmetic Flags.

- 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 20.00 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 20.01 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 20.02 to be turned ON.



MCMP

Instruction	Mnemonic	Variations	Function code	Function
MULTIPLE COMPARE	MCMP	@MCMP	019	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.

	MCMP
Symbol	 S1: First word of set 1 S2: First word of set 2 R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S1	First word of set 1	WORD	16
S2	First word of set 2	WORD	16
R	Result word	UINT	1

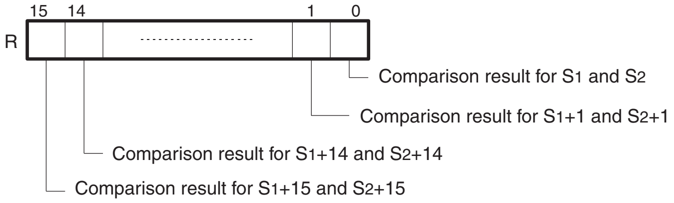
S_1: First word of set 1

S1	Comparison data 0
S1+1	Comparison data 1
	to
S1+15	Comparison data 15

S_2: First word of set 2

S2	Comparison data 0
S2+1	Comparison data 1
	to
S2+15	Comparison data 15

R: Result word



Note Specifies the beginning of the first 16-word range. S_1 and S_1 + 15 and S_2 and S_2 + 15 must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S1, S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
R												OK						

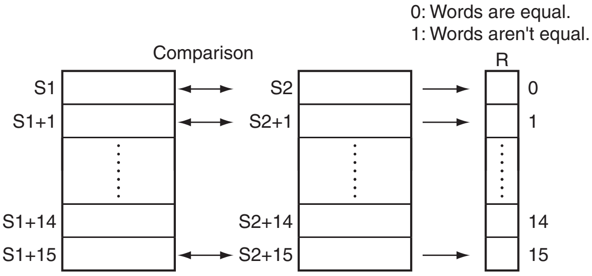
Flags

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.

Function

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.

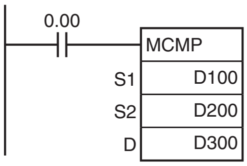


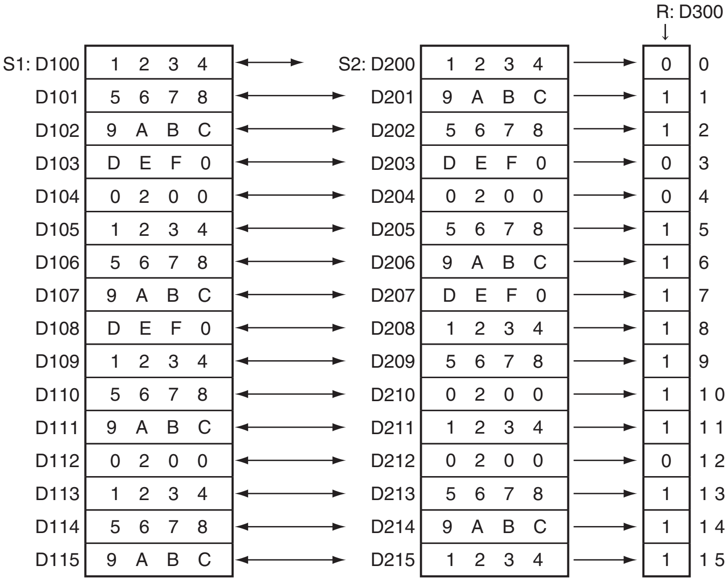
Hint

- If the contents of all 16 pairs of words are the same, the Equals Flag will turn ON after the instruction has been executed.

Example Programming

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.

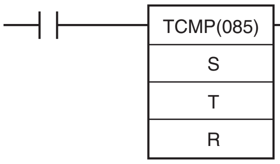




S1		S2		R: D300	Bit
S1: D100	1 2 3 4	S2: D200	1 2 3 4	0	0
D101	5 6 7 8	D201	9 A B C	1	1
D102	9 A B C	D202	5 6 7 8	1	2
D103	D E F 0	D203	D E F 0	0	3
D104	0 2 0 0	D204	0 2 0 0	0	4
D105	1 2 3 4	D205	5 6 7 8	1	5
D106	5 6 7 8	D206	9 A B C	1	6
D107	9 A B C	D207	D E F 0	1	7
D108	D E F 0	D208	1 2 3 4	1	8
D109	1 2 3 4	D209	5 6 7 8	1	9
D110	5 6 7 8	D210	0 2 0 0	1	10
D111	9 A B C	D211	1 2 3 4	1	11
D112	0 2 0 0	D212	0 2 0 0	0	12
D113	1 2 3 4	D213	5 6 7 8	1	13
D114	5 6 7 8	D214	9 A B C	1	14
D115	9 A B C	D215	1 2 3 4	1	15

TCMP

Instruction	Mnemonic	Variations	Function code	Function
TABLE COMPARE	TCMP	@TCMP	085	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.

Symbol	TCMP
	 S: Source data T: First word of table R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

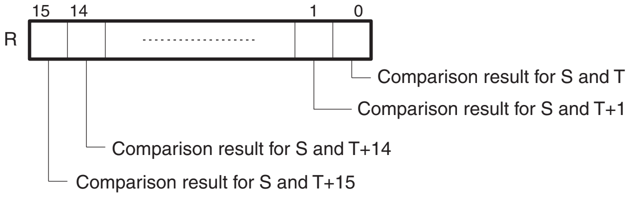
Operands

Operand	Description	Data type	Size
S	Source data	WORD	1
T	First word of table	WORD	16
R	Result word	UINT	1

T: First word of table

T	Comparison data 0
T+1	Comparison data 1
to	to
T+15	Comparison data 15

R: Result word



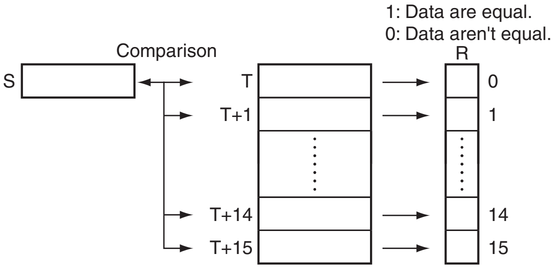
● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
T											---	---						
R												OK						

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.

Function

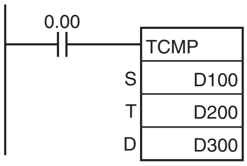


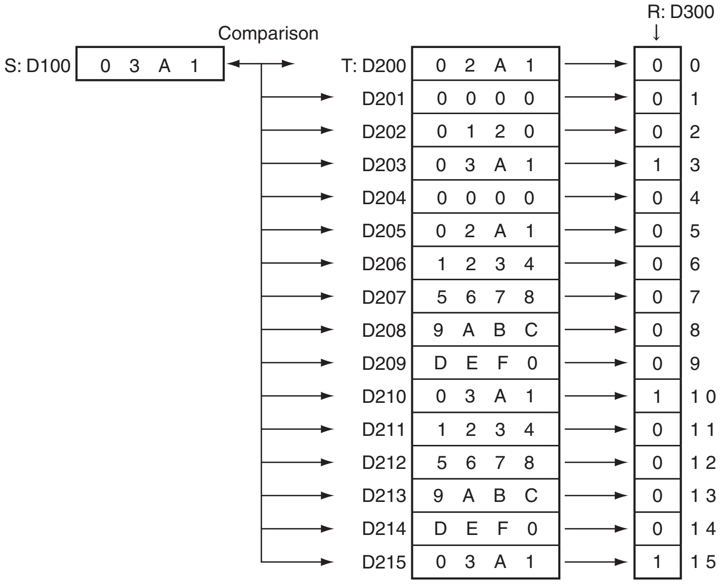
TCMP(085) compares the source data (S) to each of the 16 words T through T+15 and turns ON the corresponding bit in word R when the data **are** equal. Bit n of R is turned ON if the content of T+n is equal to S and it is turned OFF if they are not equal.

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.

Example Programming

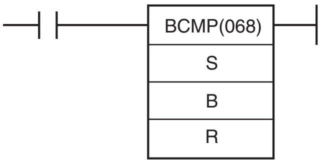
When CIO 0.00 is ON in the following example, TCMP(085) compares the content of D100 with the contents of words D200 through D215 and turns ON the corresponding bits in D300 when the contents are equal or OFF when the contents are not equal.





BCMP

Instruction	Mnemonic	Variations	Function code	Function
BLOCK COMPARE	BCMP	@BCMP	068	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.

	BCMP
Symbol	 S: Source data B: First word of block R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

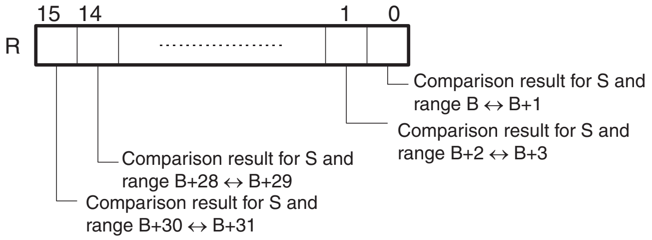
Operands

Operand	Description	Data type	Size
S	Source data	WORD	1
B	First word of block	WORD	32
R	Result word	UINT	1

B: First word of block

B	Lower limit value 0
B+1	Upper limit value 0
B+2	Lower limit value 1
B+3	Upper limit value 1
⸾	⸾
B+30	Lower limit value 15
B+31	Upper limit value 15

R: Result word



● Operand Specifications

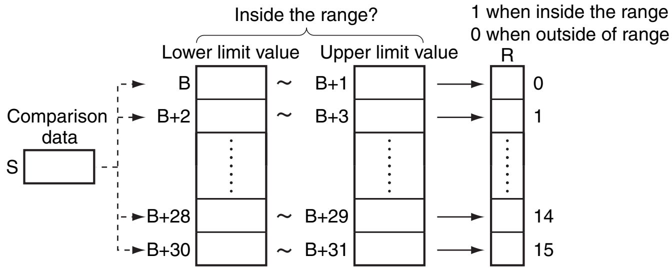
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
B											---	---						
D												OK						

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.

Function

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. If S is out of any these ranges, the corresponding bit in R is turned OFF.

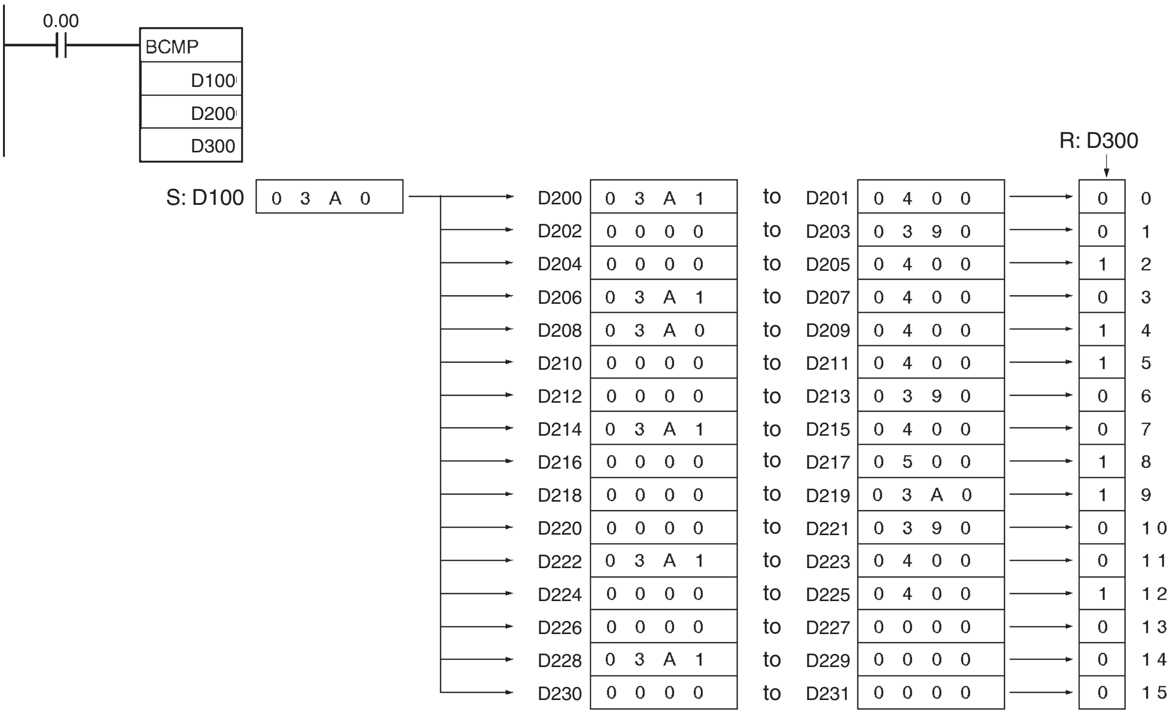


For example, bit 00 of R is turned ON if S is within the first range (B ≤ S ≤ B+1), bit 01 of R is turned ON if S is within the second range (B+2 ≤ S ≤ B+3), ..., and bit 15 of R is turned ON if S is within the fifteenth range (B+30 ≤ S ≤ B+31). All other bits in R are turned OFF.

Note 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 Programming

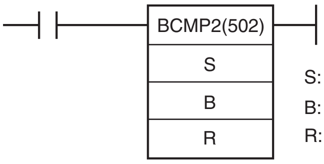
When CIO 0.00 is ON in the following example, BCMP(068) compares the content of D100 with the 16 ranges defined in D200 through D231 and turns ON the corresponding bits in D300 when S is within the range or OFF when S is not within the range.



Lower limit	Value		Upper limit	Value	R: D300	Bit
D200	0 3 A 1	to	D201	0 4 0 0	0	0
D202	0 0 0 0	to	D203	0 3 9 0	0	1
D204	0 0 0 0	to	D205	0 4 0 0	1	2
D206	0 3 A 1	to	D207	0 4 0 0	0	3
D208	0 3 A 0	to	D209	0 4 0 0	1	4
D210	0 0 0 0	to	D211	0 4 0 0	1	5
D212	0 0 0 0	to	D213	0 3 9 0	0	6
D214	0 3 A 1	to	D215	0 4 0 0	0	7
D216	0 0 0 0	to	D217	0 5 0 0	1	8
D218	0 0 0 0	to	D219	0 3 A 0	1	9
D220	0 0 0 0	to	D221	0 3 9 0	0	10
D222	0 3 A 1	to	D223	0 4 0 0	0	11
D224	0 0 0 0	to	D225	0 4 0 0	1	12
D226	0 0 0 0	to	D227	0 0 0 0	0	13
D228	0 3 A 1	to	D229	0 0 0 0	0	14
D230	0 0 0 0	to	D231	0 0 0 0	0	15

BCMP2

Instruction	Mnemonic	Variations	Function code	Function
EXPANDED BLOCK COMPARE	BCMP2	@BCMP2	502	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.

	BCMP
Symbol	 S: Source data B: First word of block R: First result word

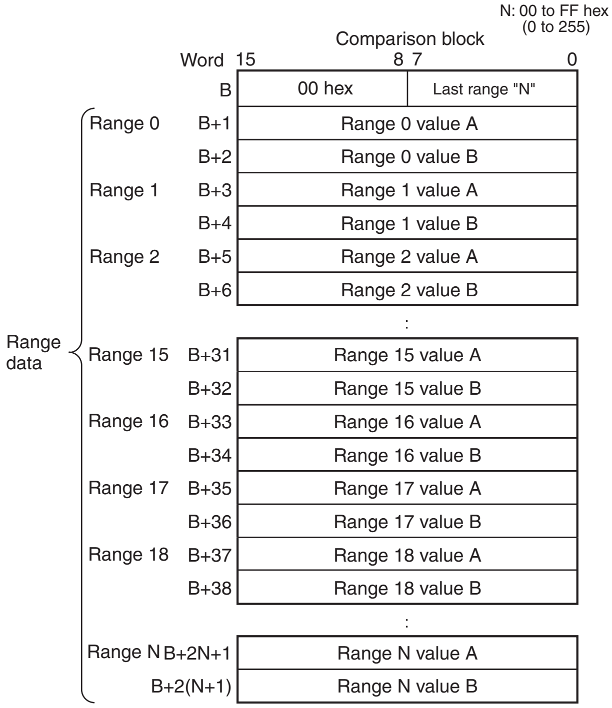
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source data	WORD	1
B	First word of block	WORD	Variable
R	Result word	WORD	Variable

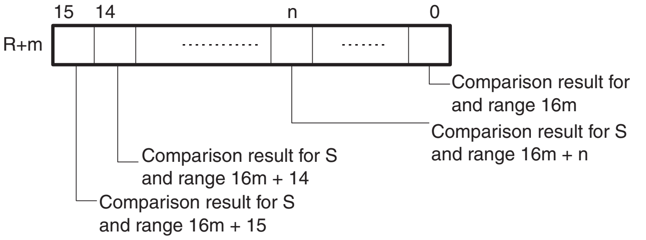
B: First word of block



	Word	15 … 8	7 … 0
	B	00 hex	Last range "N"
Range 0	B+1	Range 0 value A	
	B+2	Range 0 value B	
Range 1	B+3	Range 1 value A	
	B+4	Range 1 value B	
Range 2	B+5	Range 2 value A	
	B+6	Range 2 value B	
		:	
Range 15	B+31	Range 15 value A	
	B+32	Range 15 value B	
Range 16	B+33	Range 16 value A	
	B+34	Range 16 value B	
Range 17	B+35	Range 17 value A	
	B+36	Range 17 value B	
Range 18	B+37	Range 18 value A	
	B+38	Range 18 value B	
		:	
Range N	B+2N+1	Range N value A	
	B+2(N+1)	Range N value B	

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
B											---	---						
R																		

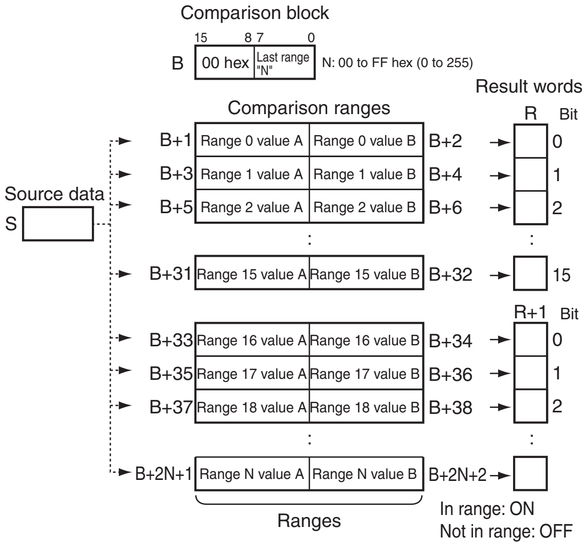
Flags

Name	Label	Operation
Error Flag	P_ER	OFF

Function

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.



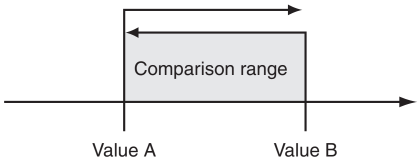
● Number of Ranges

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



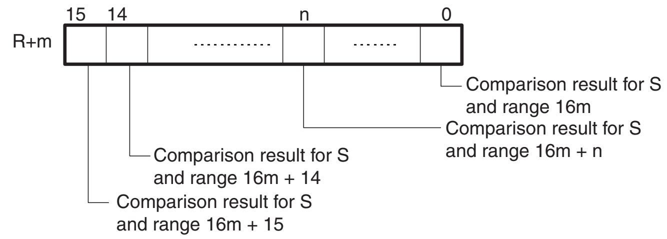
- If Value A ≤ Value B
  Then, Comparison range ≤ Value B and Value A ≤ Comparison range

- **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

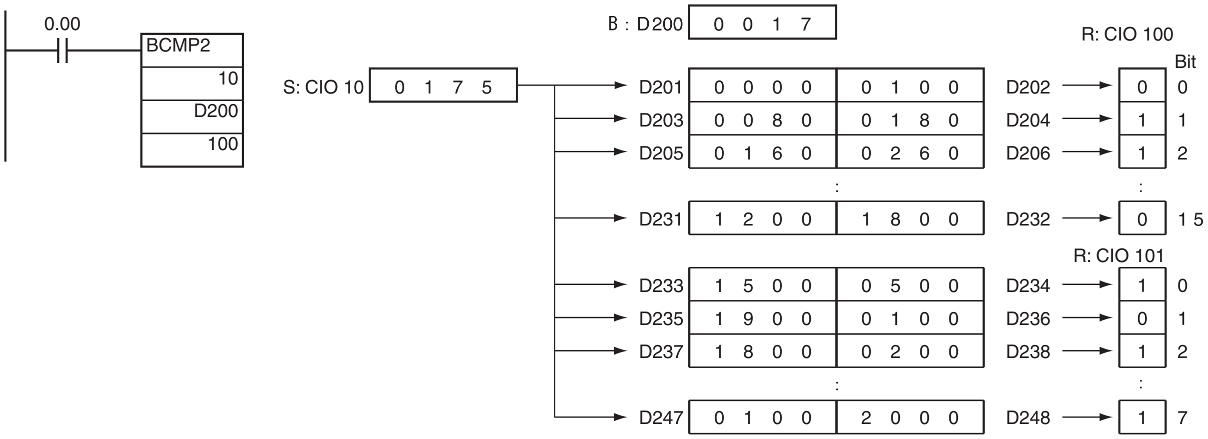
The results are output to corresponding bits in word R. If there are more than 16 comparison ranges, consecutive words following R will be used.

The maximum number of result words is 16, i.e., m equals 0 to 15.



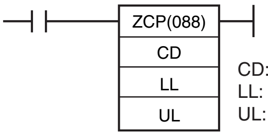
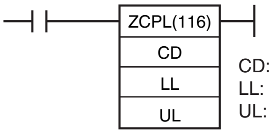
Example Programming

When CIO 0.00 is ON in the following example, BCMP2(502) compares the content of CIO 10 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 100 and CIO 101 when S is within the range and OFF when S is not within the range. For example, if the source data in CIO 10 is in the range defined by D201 and D202, then bit 00 of CIO 100 is turned ON and if it in not in the range, then bit 00 of CIO 100 is turned OFF. Likewise, the source data in CIO 10 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 100, bit 7 in CIO 101, and the other bits in the result words are manipulated according to the results of comparison.



ZCP/ZCPL

Instruction	Mnemonic	Variations	Function code	Function
AREA RANGE COMPARE	ZCP	---	088	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.
DOUBLE AREA RANGE COMPARE	ZCPL	---	116	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.

	ZCP	ZCPL
Symbol	 CD: Comparison data LL: Lower limit of range UL: Upper limit of range	 CD: First word of comparison data LL: First word of lower limit UL: First word of upper limit

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		CMP	CMPL	CMP	CMPL
CD	ZCP: Comparison data (one word of data) ZCPL: Comparison data (two words of data)	UINT	UDINT	1	2
LL	ZCP: Low limit ZCPL: Low limit leftmost word number	UINT	UDINT	1	2
UL	ZCP: High limit ZCPL: First word of upper limit	UINT	UDINT	1	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ZCP: CD, LL, UL	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
ZCPL: CD, LL, UL												---	OK					

Flags

Name	Label	Operation	
		ZCP	ZCPL
Error Flag	P_ER	ON if LL > UL.	ON if LL+1, LL > UL+1, UL.
Greater Than Flag	P_GT	• ON if CD > UL. • OFF in all other cases.	• ON if CD > UL+1, UL. • OFF in all other cases.
Greater Than or Equal Flag	P_GE	• CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit: Held • CJ2H (unit version 1.3 or later) or CJ2M CPU Unit: • ON if CD ≤ LL • OFF in all other cases.	• CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit: Held • CJ2H (unit version 1.3 or later) or CJ2M CPU Unit: • ON if CD+1, CD ≤ LL+1, LL • OFF in all other cases.
Equal Flag	P_EQ	• ON if LL ≤ CD ≤ UL. • OFF in all other cases.	• ON if LL+1, LL ≤ CD+1, CD ≤ UL+1, UL. • OFF in all other cases.

Name	Label	Operation	
		ZCP	ZCPL

Not Equal Flag	P_NE	• CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit: Held
		• CJ2H (unit version 1.3 or later) or CJ2M CPU Unit:
		  • ON if CD < LL.
		  • ON if CD > UL.
		  • OFF in all other cases.
			• CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit: Held
			• CJ2H (unit version 1.3 or later) or CJ2M CPU Unit:
			  • ON if CD+1, CD < LL+1, LL.
			  • ON if CD+1, CD > UL+1, UL.
			  • OFF in all other cases.

Less Than Flag	P_LT	• ON if CD < LL. • OFF in all other cases.	• ON if CD+1, CD < LL+1, LL. • OFF in all other cases.

Less Than or Equal Flag	P_LE	• CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit: Held
		• CJ2H (unit version 1.3 or later) or CJ2M CPU Unit:
		  • ON if CD ≤ UL.
		  • OFF in all other cases.
			• CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit: Held
			• CJ2H (unit version 1.3 or later) or CJ2M CPU Unit:
			  • ON if LL+1, LL ≤ CD+1, CD.
			  • OFF in all other cases.

Negative Flag	P_N	Held	Held

Function

● ZCP

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.)

When S > T2 as shown below, the > flag turns ON.
When T1 ≤ S ≤ T2, the = flag turns ON. When S < T1, the < flag turns ON.

ZCP(088)Result	Flag status		
	>	=	<
CD > UL	ON	OFF	OFF
CD = UL	OFF	ON	
LL < CD < UL			
CD = LL			
CD < LL		OFF	ON

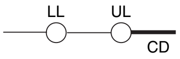
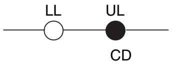
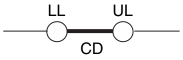
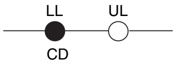
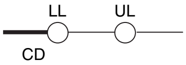
● ZCPL

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.)

When S+1,S > T2+1,T2 as shown below, the > flag turns ON.
When T1+1,T1 ≤ S+1, S ≤ T2+1, T2, the = flag turns ON.
When S+1, S < T1+1, T1, the < flag turns ON.

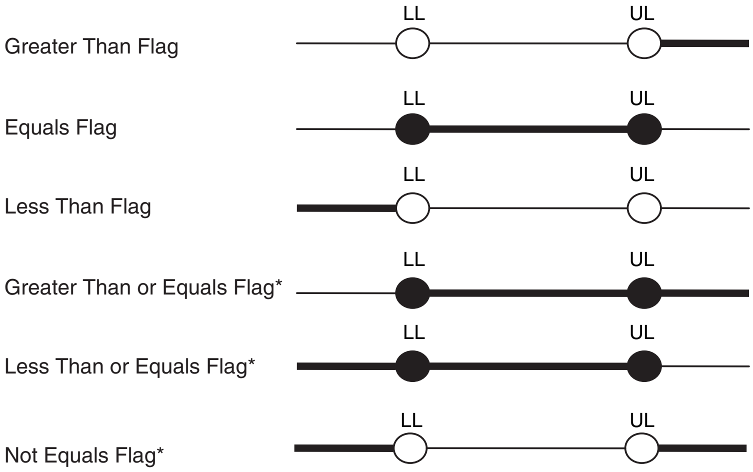
ZCPL(116)Result	Flag status		
	>	=	<
CD+1, CD > UL+1, UL	ON	OFF	OFF
CD+1, CD = UL+1, UL	OFF	ON	
LL+1, LL < CD+1, CD < UL+1, UL			
CD+1, CD = LL+1, LL			
CD+1, CD < LL+1, LL		OFF	ON

Comparison Result and Flag Operation

Result	Greater Than Flag	Equals Flag	Less Than Flag	Greater Than or Equals Flag* (AND of Greater Than Flag and Equals Flag)	Less Than or Equals Flag* (AND of Less Than Flag and Equals Flag)	Not Equals Flag* (AND of Greater Than Flag and Less Than Flag)
CD > UL 	ON	OFF	OFF	ON	OFF	ON
CD = UL 	OFF	ON			ON	OFF
LL < CD < UL 						
CD = LL 						
CD < LL 		OFF	ON	OFF		ON

* These flags do not turn ON and OFF in the CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, and CS1D CPU Units. They turn ON and OFF only in CJ2H (unit version 1.3 or later) and CJ2M CPU Units.

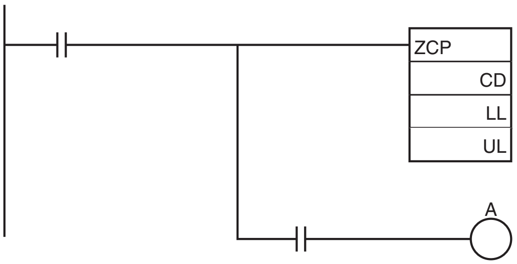
Each flag shown below is ON when CD is on the bold lines or the filled circles.



* These flags do not turn ON and OFF in the CJ2H (unit version 1.0 to 1.2), CS1-H, CJ1-H, CJ1M, and CS1D CPU Units.

Precautions

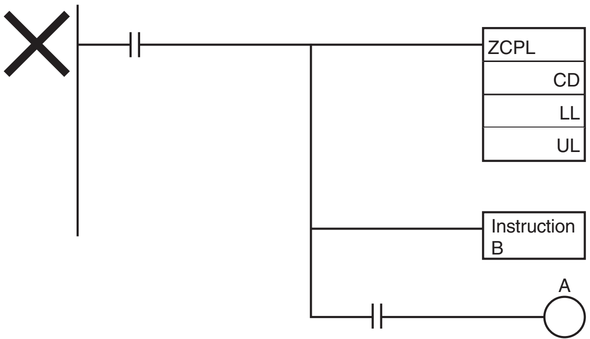
- When ZCP(088)/ZCPL(116) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls ZCP(088)/ZCPL(116), as shown in the following diagram.



Arithmetic Flag
(Example: Equal Flag)

In this case, the Equals Flag and output A will be turned ON when LL ≤ CD ≤ UL.

- Do not program another instruction between ZCP(088)/ZCPL(116) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag.

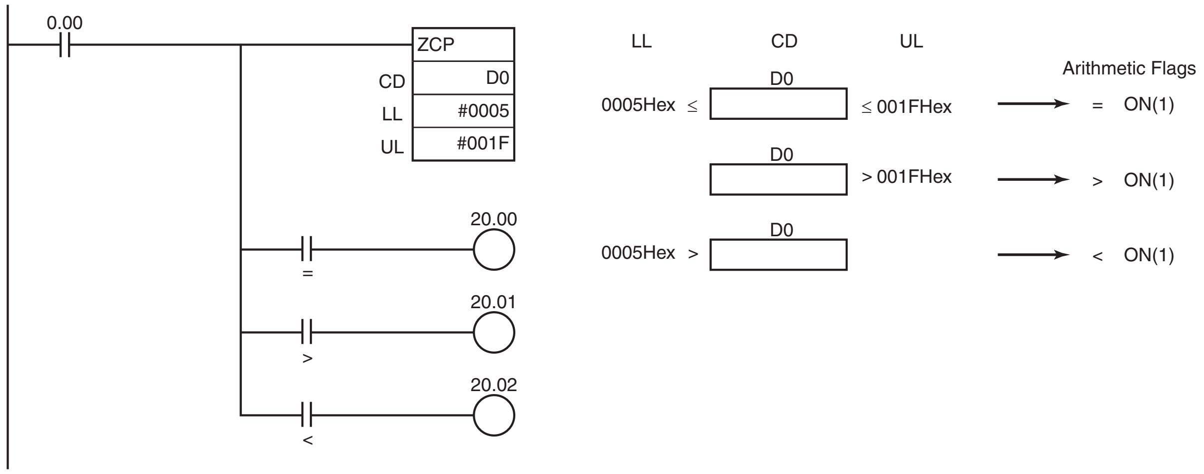


Arithmetic Flag
(Example: Equal Flag)

In this case, the results of instruction B might change the results of ZCP(088)/ZCPL(116).

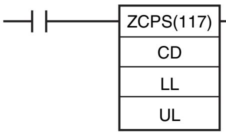
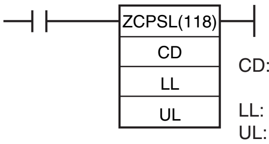
Example Programming

- 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 20.00 is turned ON if 0005 hex ≤ content of D0 ≤ 001F hex.
  CIO 20.01 is turned ON if the content of D0 > 001F hex.
  CIO 20.02 is turned ON if the content of D0 < 0005 hex.



ZCPS/ZCPSL

Instruction	Mnemonic	Variations	Function code	Function
SIGNED AREA RANGE COMPARE	ZCPS	---	117	Compares the 16-bit signed binary value in CD (word contents or constant) to the range defined by LL and UL and outputs the results to the Arithmetic Flags.
DOUBLE SIGNED AREA RANGE COMPARE	ZCPSL	---	118	Compares the 32-bit signed binary value in CD and CD+1 (word contents or constant) to the range defined by LL+1, LL and UL+1, UL and outputs the results to the Arithmetic Flags.

	ZCPS	ZCPSL
Symbol	 CD: Comparison data (one word) LL: Lower limit of range UL: Upper limit of range	 CD: First word of comparison data (two words) LL: First word of lower limit UL: First word of upper limit

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ZCPS	ZCPSL	ZCPS	ZCPSL
CD	ZCPS: Comparison data (one word of data) ZCPSL: Comparison data (two words of data)	INT	DINT	1	2
LL	ZCPS: Lower limit ZCPSL: First word of lower limit	INT	DINT	1	2
UL	ZCPS: Upper limit ZCPSL: First word of upper limit	INT	DINT	1	2

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ZCPS: CD, LL, UL	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
ZCPSL: CD, LL, UL												---	OK					

Flags

Name	Label	Operation	
		ZCP	ZCPL
Error Flag	P_ER	ON if LL > UL.	ON if LL+1, LL > UL+1, UL.
Greater Than Flag	P_GT	• ON if CD > UL. • OFF in all other cases.	• ON if CD+1, CD > UL+1, UL. • OFF in all other cases.
Greater Than or Equal Flag	P_GE	• ON if LL ≤ CD. • OFF in all other cases.	• ON if LL+1, LL ≤ CD+1, CD. • OFF in all other cases.
Equal Flag	P_EQ	• ON if LL ≤ CD ≤ UL. • OFF in all other cases.	• ON if LL+1, LL ≤ CD+1, CD ≤ UL+1, UL. • OFF in all other cases.
Not Equal Flag	P_NE	• ON if CD < LL. • ON if CD > UL. • OFF in all other cases.	• ON if CD+1, CD < LL+1, LL. • ON if CD+1, CD > UL+1, UL. • OFF in all other cases.

Name	Label	Operation	
		ZCP	ZCPL
Less Than Flag	P_LT	• ON if CD < LL. • OFF in all other cases.	• ON if CD+1, CD < LL+1, LL. • OFF in all other cases.
Less Than or Equal Flag	P_LE	• ON if CD ≤ UL. • OFF in all other cases.	• ON if CD+1, CD ≤ UL+1, UL. • OFF in all other cases.
Negative Flag	P_N	Held	Held

Function

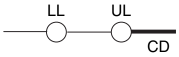
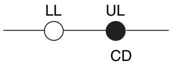
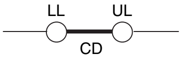
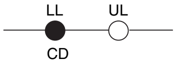
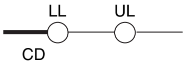
● ZCPS

ZCPS(117) compares the 16-bit signed binary data in CD with the range defined by LL and UL and outputs the result to the Arithmetic Flags.

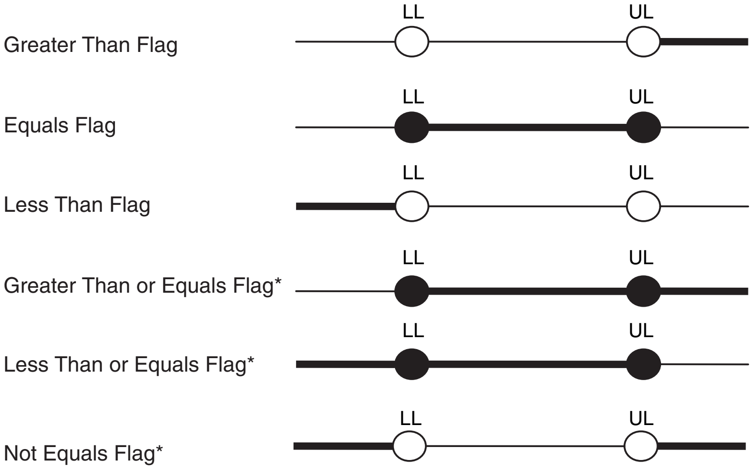
● ZCPSL

ZCPSL(118) 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 Arithmetic Flags.

Comparison Result and Flag Operation

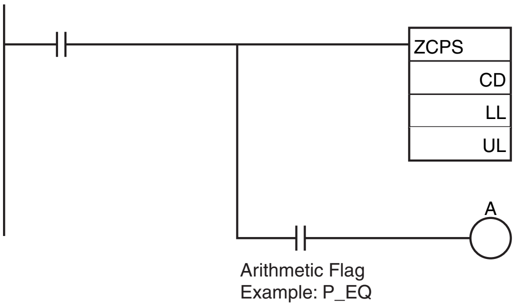
Result	Greater Than Flag	Equals Flag	Less Than Flag	Greater Than or Equals Flag* (AND of Greater Than Flag and Equals Flag)	Less Than or Equals Flag* (AND of Less Than Flag and Equals Flag)	Not Equals Flag* (AND of Greater Than Flag and Less Than Flag)
CD > UL 	ON	OFF	OFF	ON	OFF	ON
CD = UL 	OFF	ON			ON	OFF
LL < CD < UL 						
CD = LL 						
CD < LL 		OFF	ON	OFF		ON

Each flag shown below is ON when CD is on the bold lines or the filled circles.



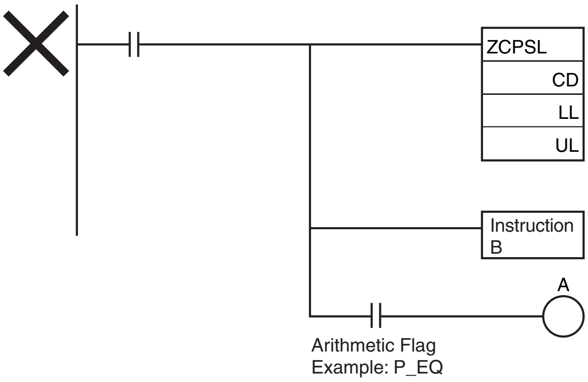
Precautions

- When ZCPS(117)/ZCPSL(118) 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 ZCPS(117)/ZCPSL(118), as shown in the following diagram.



If LL ≤ CD ≤ UL, P_EQ turns ON
and A turns ON.

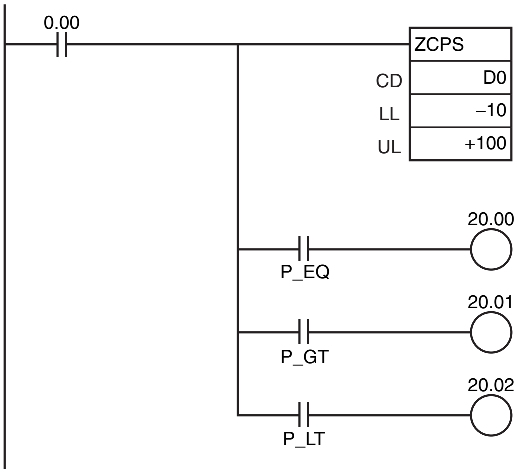
- Do not program another instruction between ZCPS(117)/ZCPSL(118) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag.

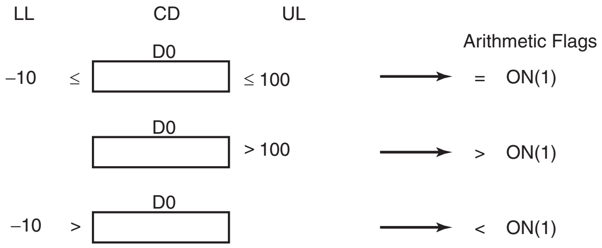


In this case, the results of instruction B
might change the results of ZCPS(117)/ZCPSL(118).

Example Programming

- When CIO 0.00 is ON in the following example, the 16-bit signed binary data in D0 is compared to the range FFF6 to 0064 hex (–10 to 100 decimal) and the result is output to the Arithmetic Flags.
- CIO 20.00 is turned ON if –10 decimal ≤ content of D0 ≤ 100 decimal. CIO 20.01 is turned ON if the content of D0 > 100 decimal. CIO 20.02 is turned ON if the content of D0 < –10 decimal.

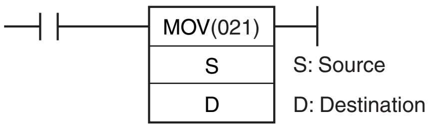
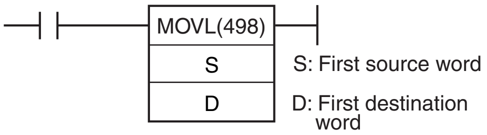




Data Movement Instructions

MOV/MOVL

Instruction	Mnemonic	Variations	Function code	Function
MOVE	MOV	@MOV, !MOV, !@MOV	021	Transfers a word of data to the specified word.
DOUBLE MOVE	MOVL	@MOVL	498	Transfers two words of data to the specified words.

	MOV	MOVL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		MOV	MOVL	MOV	MOVL
S	MOV: Source MOVL: First source word	WORD	DWORD	1	2
D	MOV: Destination MOVL: First destination word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
MOV	S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	D											---							
MOVL	S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	OK	---	---	---	---
	D											---							

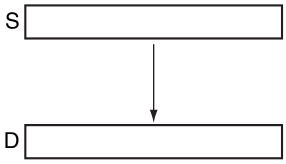
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equal Flag	=	• ON if the data being transferred (D) is 0. • OFF in all other cases.
Negative Flag	N	• ON if the leftmost bit of the data being transferred (D) is 1. • OFF in all other cases.

Function

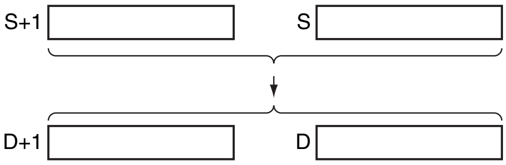
● MOV

Transfers S to D. If S is a constant, the value can be used for a data setting.



● MOVL

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.



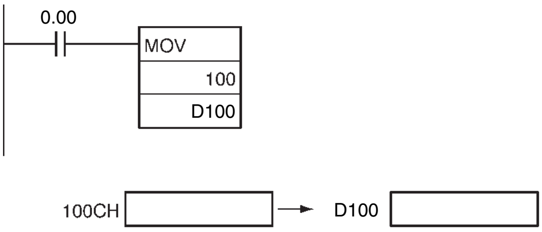
Precautions

MOV(021) has an immediate refreshing variation (!MOV(021)). An 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 unless the bits are allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.

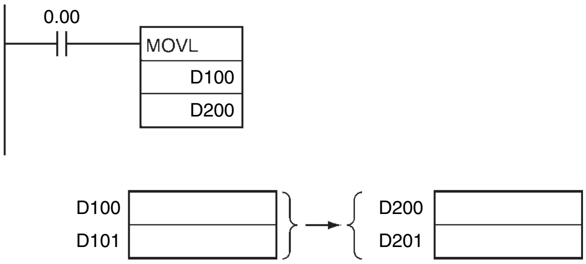
When external input is specified for S, the value of S will be in-refreshed when the instruction is executed and transferred to D. When external output is specified for D, the value of S will be transferred to D and immediately out-refreshed when the instruction is executed. It is also possible to in-refresh S and out-refresh D at the same time.

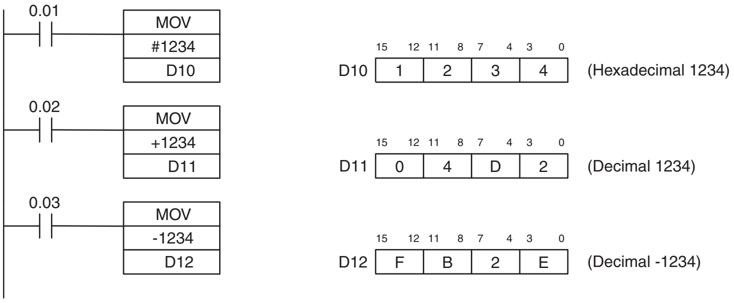
Example Programming

When CIO 0.00 is ON in the following example, the content of CIO 100 is copied to D100.



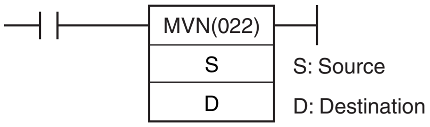
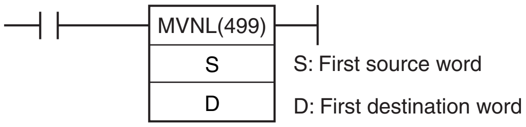
When CIO 0.00 is ON in the following example, the content of D101 and D100 are copied to D201 and D200.





MVN/MVNL

Instruction	Mnemonic	Variations	Function code	Function
MOVE NOT	MVN	@MVN	022	Transfers the complement of a word of data to the specified word.
DOUBLE MOVE NOT	MVNL	@MVNL	499	Transfers the complement of two words of data to the specified words.

	MVN	MVNL
Symbol	 MVN(022) S: Source D: Destination	 MVNL(499) S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		MVN	MVNL	MVN	MVNL
S	MVN: Source MVNL: First source word	WORD	DWORD	1	2
D	MVN: Destination MVNL: First destination word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
MVN	S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	D											---							
MVNL	S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	D											---							

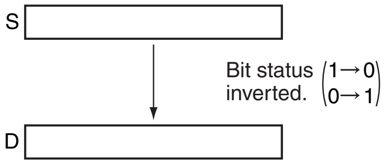
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equal Flag	=	• ON if the data being transferred (D) is 0. • OFF in all other cases.
Negative Flag	N	• ON if the leftmost bit of the data being transferred (D) is 1. • OFF in all other cases.

Function

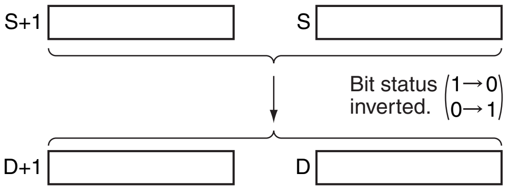
MVN

MVN(022) inverts the bits in S and transfers the result to D. The content of S is left unchanged.



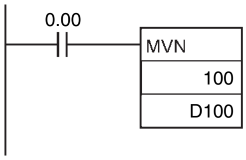
MVNL

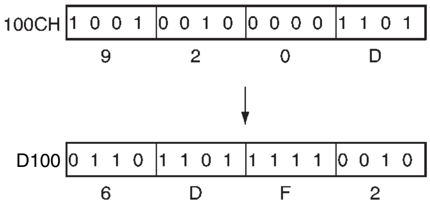
MVNL(499) inverts the bits in S+1 and S and transfers the result to D+1 and D. The contents of S+1 and S are left unchanged.



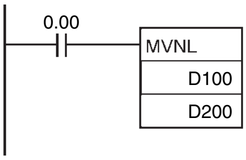
Example Programming

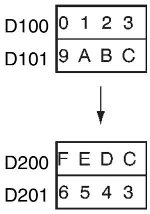
When CIO 0.00 is ON in the following example, the status of the bits in CIO 100 is inverted and the result is copied to D100.





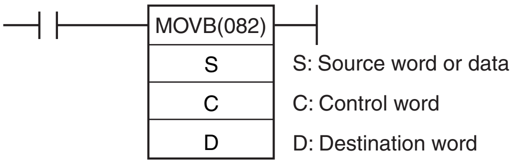
When CIO 0.00 is ON in the following example, the status of the bits in D101 and D100 are inverted and the result is copied to D201 and D200. (The original contents of D101 and D100 are left unchanged.)





MOVB

Instruction	Mnemonic	Variations	Function code	Function
MOVE BIT	MOVB	@MOVB	082	Transfers the specified bit.

	MOVB
Symbol	

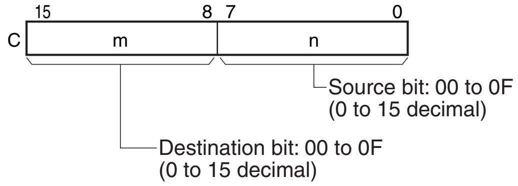
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word or data	WORD	1
C	Control word	UINT	1
D	Destination word	WORD	1

C: Control Word



● Operand Specifications

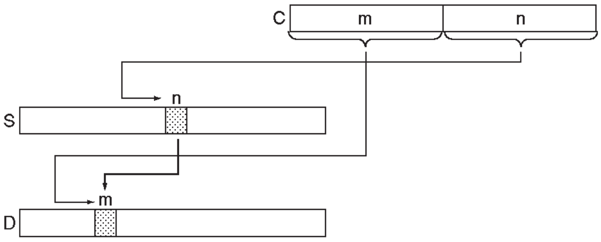
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C																		
D											---							

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.

Function

MOVB(082) copies the specified bit (n) from S to the specified bit (m) in D.



Hint

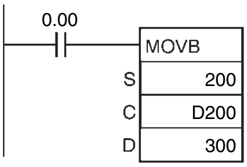
The same word can be specified for both S and D to copy a bit within a word.

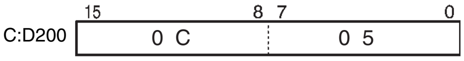
Precautions

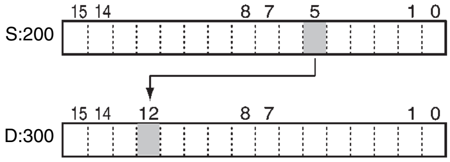
The other bits in the destination word are left unchanged.

Example Programming

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 0C05.

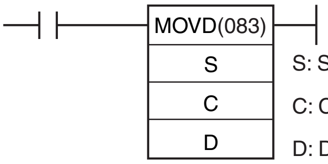






MOVD

Instruction	Mnemonic	Variations	Function code	Function
MOVE DIGIT	MOVD	@MOVD	083	Transfers the specified digit or digits. (Each digit is made up of 4 bits.)

	MOVB
Symbol	 S: Source word or data C: Control word D: Destination word

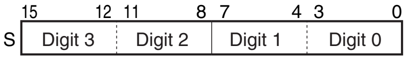
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

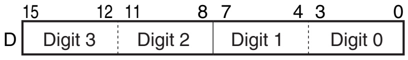
Operand	Description	Data type	Size
S	Source word or data	WORD	1
C	Control word	UINT	1
D	Destination word	UINT	1

S: Source Word



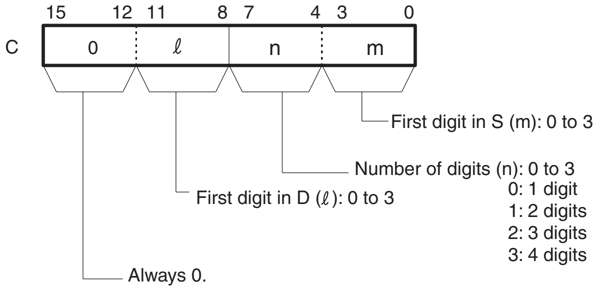
The source digits are read from right to left, wrapping back to the rightmost digit (digit 0) if necessary.

D: Destination Word



The destination digits are written from right to left, wrapping back to the rightmost digit (digit 0) if necessary.

C: Control Word



● Operand Specifications

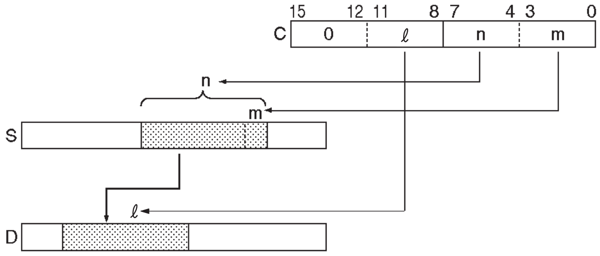
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C																		
D											---							

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.

Function

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.



Hint

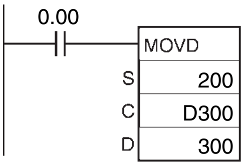
The same word can be specified for both S and D to copy a bit within a word.

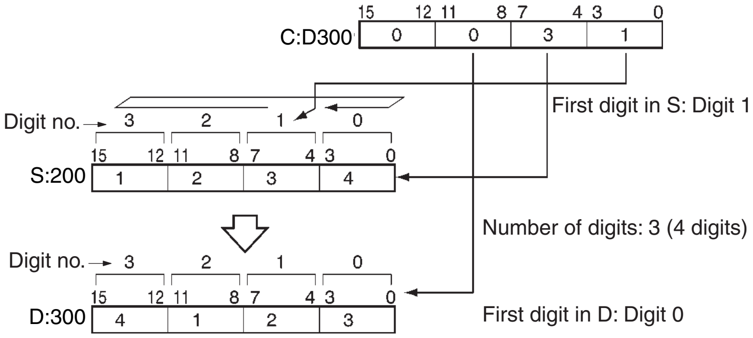
Precautions

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.

Example Programming

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 of CIO 300, in accordance with the control word's value of 31.





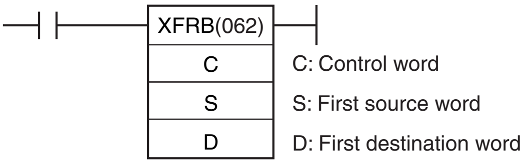
Note After reading the leftmost digit of S (digit 3), MOVD(083) wraps to the rightmost digit (digit 0).

● Example of transferring multiple digits

The following diagram shows examples of data transfers for various values of C.

XFRB

Instruction	Mnemonic	Variations	Function code	Function
MULTIPLE BIT TRANSFER	XFRB	@XFRB	062	Transfers the specified number of consecutive bits.

	XFRB
Symbol	

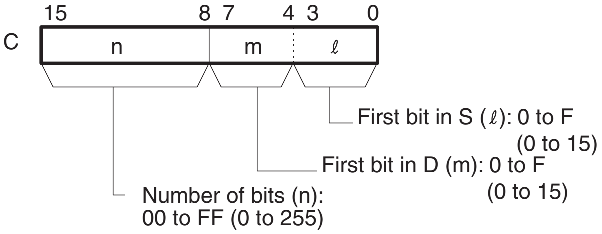
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

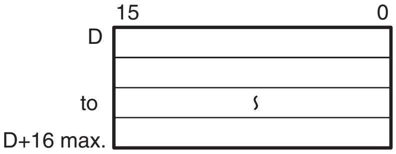
Operands

Operand	Description	Data type	Size
C	Control word	UINT	1
S	First source word	WORD	Variable
D	First destination word	WORD	Variable

C: Control Word

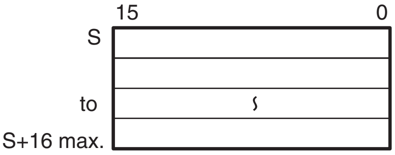


D: First destination Word



Note The source words and the destination words must be in the same data area respectively.

S: First Source Word



● Operand Specifications

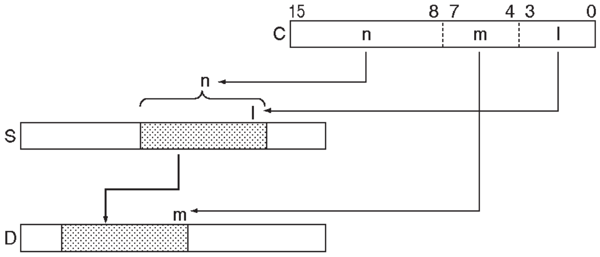
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
D																		

Flags

Name	Label	Operation
Error Flag	ER	OFF

Function

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).
The beginning bits and number of bits are specified in C, as shown in the following diagram.



Hint

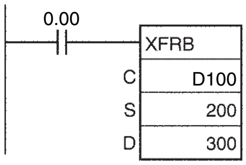
- Up to 255 bits of data can be transferred per execution of XFRB(062).
- It is possible for the source words and destination words to overlap. By transferring data overlapping several words, the data can be packed more efficiently in the data area. (This is particularly useful when handling position 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.

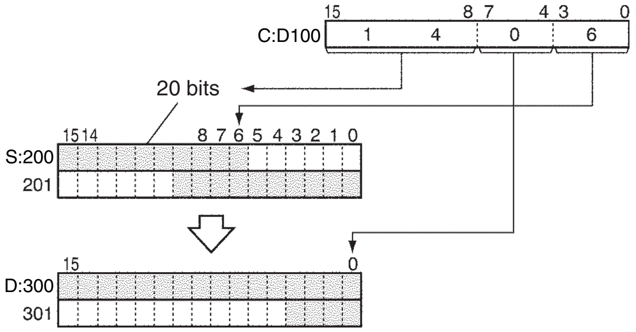
Precautions

- Be sure that the source words and destination words do not exceed the end of the data area.
- When the number of transfer bits (n of C) is 0, transfer does not take place.
- Bits in the destination words that are not overwritten by the source bits are left unchanged.

Example Programming

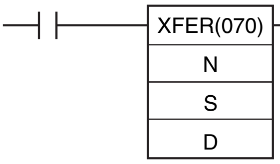
When CIO 0.00 is ON in the following example, the 20 bits beginning with CIO 200.06 are copied to the 20 bits beginning with CIO 300.





XFER

Instruction	Mnemonic	Variations	Function code	Function
BLOCK TRANSFER	XFER	@XFER	070	Transfers the specified number of consecutive words.

	XFER
Symbol	 N: Number of words S: First source word D: First destination word

Applicable Program Areas

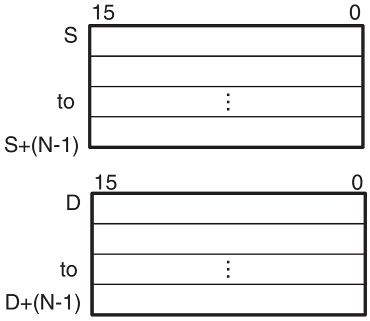
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Number of words	UINT	1
S	First source word	WORD	Variable
D	First destination word	WORD	Variable

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).



● Operand Specifications

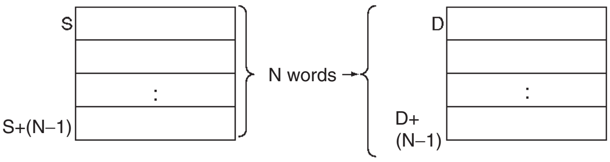
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
D																		

Flags

Name	Label	Operation
Error Flag	ER	OFF

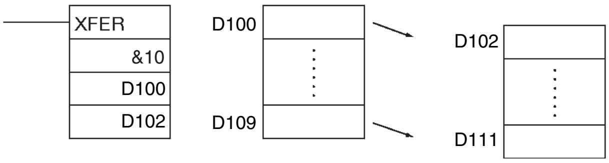
Function

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)).



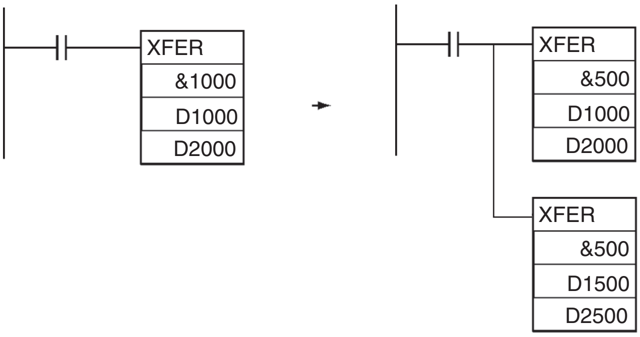
Hint

- It is possible for the source words and destination words to overlap, so XFER(070) can perform word-shift operations.
- The specified source and destination data areas can overlap (word shift).



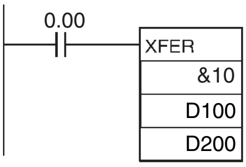
Precautions

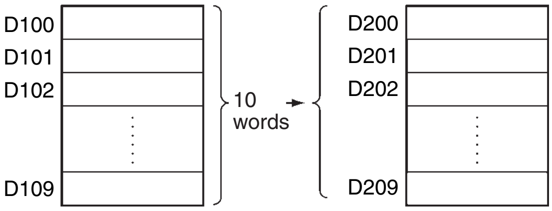
- Be sure that the source words (S to S+N-1) and destination words (D to D+N-1) do not exceed the end of the data area.
- Some time will be required to complete XFER(070) when a large number of words is being transferred. Even if an interrupt occurs, execution of this instruction will not be interrupted and execution of the interrupt task will be started after execution of XFER(070) has been completed. If power is interrupted during execution of XFER(070), execution may not be completed, i.e., all of the specified data may not be transferred. One XFER(070) instruction can be replaced with two XFER(070) instructions to help avoid this problem.



Example Programming

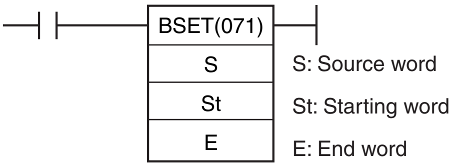
When CIO 0.00 is ON in the following example, the 10 words D100 through D109 are copied to D200 through D209.





BSET

Instruction	Mnemonic	Variations	Function code	Function
BLOCK SET	BSET	@BSET	071	Copies the same word to a range of consecutive words.

	BSET
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

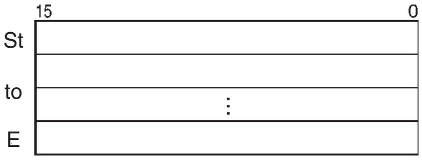
Operand	Description	Data type	Size
S	Source word	WORD	1
St	Starting word	WORD	Variable
E	End word	WORD	Variable

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
St											---	---						
E																		

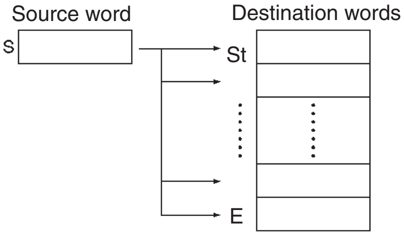
Flags

Name	Label	Operation
Error Flag	ER	• ON if St is greater than E. • OFF in all other cases.

Note Be sure that the starting word (St) and end word (E) are in the same data area and that St ≤ E.

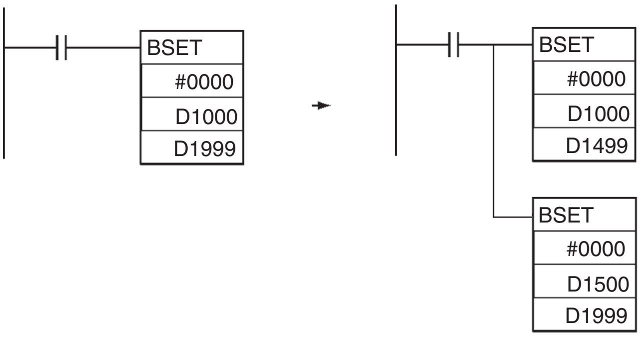
Function

BSET(071) copies the same source word (S) to all of the destination words in the range St to E.



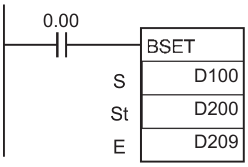
Precautions

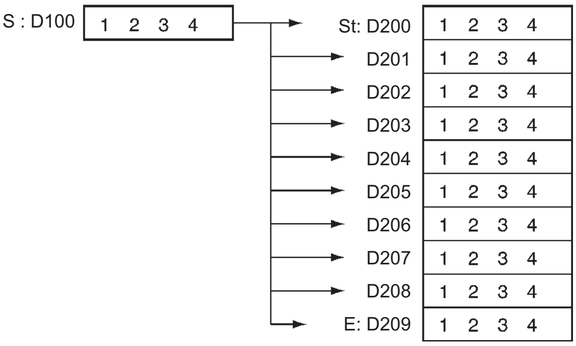
- Some time will be required to complete BSET(071) when a large number of words is being set. Even if an interrupt occurs, execution of this instruction will not be interrupted and execution of the interrupt task will be started after execution of BSET(071) has been completed. If power is interrupted during execution of BSET(071), execution may not be completed, i.e., all of the specified words may not be set. One BSET(071) instruction can be replaced with two BSET(071) instructions to help avoid this problem.



Example Programming

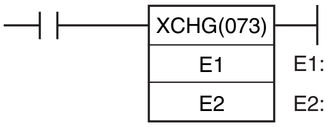
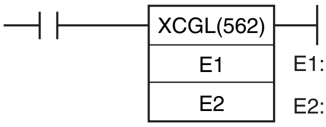
When CIO 0.00 is ON in the following example, the source data in D100 is copied to D200 through D209.





XCHG/XCGL

Instruction	Mnemonic	Variations	Function code	Function
DATA EXCHANGE	XCHG	@XCHG	073	Exchanges the contents of the two specified words.
DOUBLE DATA EXCHANGE	XCGL	@XCGL	562	Exchanges the contents of a pair of consecutive words with another pair of consecutive words.

	XCHG	XCGL
Symbol	 E1: First exchange word E2: Second exchange word	 E1: First exchange word E2: Second exchange word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		XCHG	XCGL	XCHG	XCGL
E1	First exchange word	WORD	DWORD	1	2
E2	Second exchange word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
XCHG	E1,E2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
XCGL	E1,E2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

Function

● XCHG

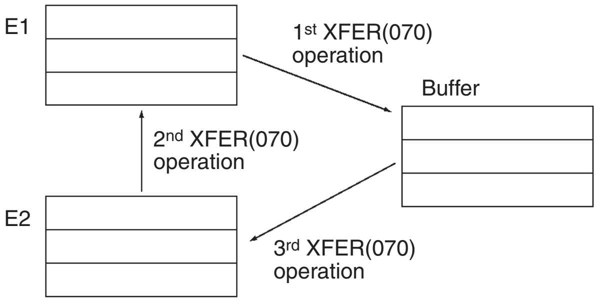
XCHG(073) exchanges the contents of E1 and E2.

● XCGL

XCHG(073) exchanges the contents of E1+1 and E1 with the contents of E2+1 and E2.

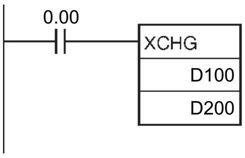
Hint

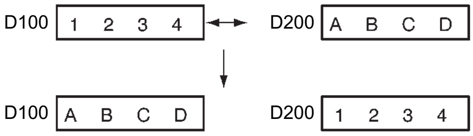
To exchange 3 or more words, use XFER(070) to transfer the words to a third set of words (a buffer) as shown in this diagram.



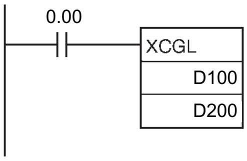
Example Programming

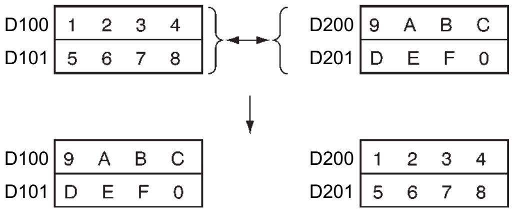
When CIO 0.00 is ON in this example, the content of D100 is exchanged with the content of D200.





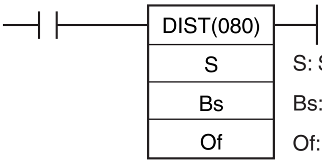
When CIO 0.00 is ON in this example, the contents of D100 and D101 are exchanged with the contents of D200 and D201.





DIST

Instruction	Mnemonic	Variations	Function code	Function
SINGLE WORD DISTRIBUTE	DIST	@DIST	080	Transfers the source word to a destination word calculated by adding an offset value to the base address.

Symbol	DIST
	 S: Source word Bs: Destination base address Of: Offset

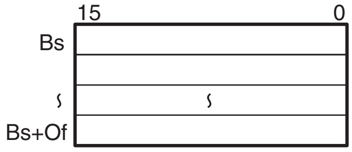
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
Bs	Destination base address	WORD	1
Of	Offset	UINT	1

Bs: Destination Base Address



Of: Offset

The offset can be any value from 0000 to FFFF (0 to 65,535 decimal).

Note Bs and Bs+Of must be in the same data area.

● Operand Specifications

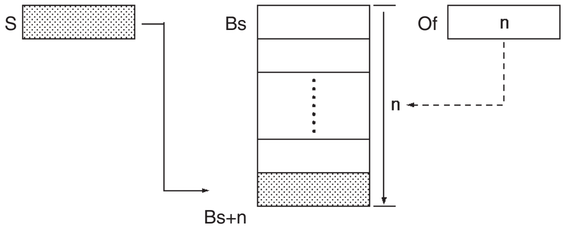
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
Bs											---	---						
Of											OK	OK						

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

Function

DIST(080) copies S to the destination word calculated by adding Of to Bs.



Hint

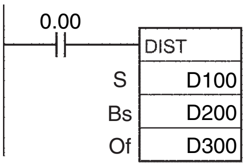
The same DIST(080) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.

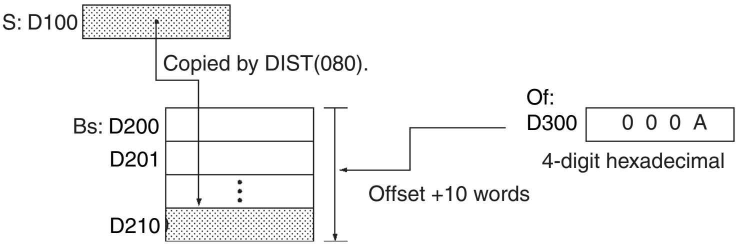
Precautions

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 Programming

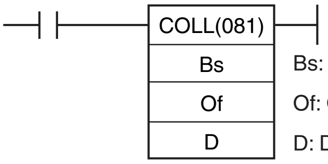
When CIO 0.00 is ON in this example, the contents of D100 will be copied to D210 (D200 + 10) if the contents of D300 is 10 (0A hexadecimal). The contents of D100 can be copied to other words by changing the offset in D300.





COLL

Instruction	Mnemonic	Variations	Function code	Function
DATA COLLECT	COLL	@COLL	081	Transfers the source word (calculated by adding an offset value to the base address) to the destination word.

	COLL
Symbol	 Bs: Source base address Of: Offset D: Destination word

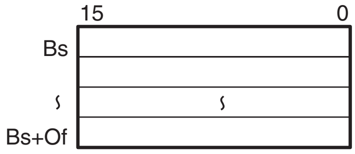
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
Bs	Source base address	WORD	1
Of	Offset	WORD	1
D	Destination word	WORD	1

Bs: Source Base Address



Of: Offset

The offset can be any value from 0000 to FFFF (0 to 65,535 decimal).

Note Bs and Bs+Of must be in the same data area.

● Operand Specifications

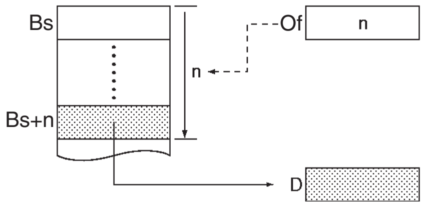
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
Bs	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
Of											OK	OK						
D											---							

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

Function

COLL(081) copies the source word (calculated by adding Of to Bs) to the destination word.



Hint

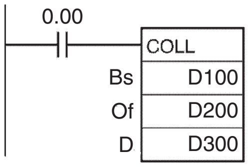
The same COLL(081) instruction can be used to collect data from various source words in the data area by changing the value of Of.

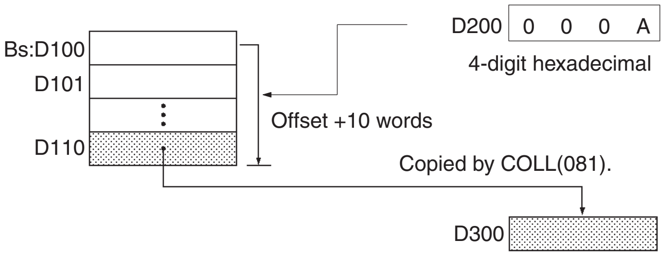
Precautions

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 Programming

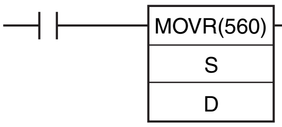
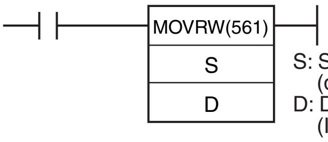
When CIO 0.00 is ON in the following example, the contents of D110 (D100 + 10) will be copied to D300 if the content of D200 is 10 (0A hexadecimal). The contents of other words can be copied to D300 by changing the offset in D200.





MOVR/MOVRW

Instruction	Mnemonic	Variations	Function code	Function
MOVE TO REGISTER	MOVR	@MOVR	560	Sets the PLC memory address of the specified word, bit, or timer/counter Completion Flag in the specified Index Register.
MOVE TIMER/COUNTER PV TO REGISTER	MOVRW	@MOVRW	561	Sets the PLC memory address of the specified timer or counter's PV in the specified Index Register.

	MOVR	MOVRW
Symbol	 S: Source (desired word or bit) D: Destination (Index Register)	 S: Source (desired TC number) D: Destination (Index Register)

Applicable Program Areas

● MOVR

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

● MOVRW

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK	OK	Not allowed

Operands

Operand	Description	Data type		Size
		MOVR	MOVRW	
S	MOVR: Source (desired word or bit) MOVRW: Source (desired TC number)	BOOL	UINT	1
D	Destination	WORD	WORD	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
MOVR	S	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	---	---	---	OK	---	---	---
	D	---	---	---	---	---	---	---	---					OK		---			
MOVRW	S	---	---	---	---	OK	OK	---	---	---	---	---	---	---	---	---	---	---	---
	D					---	---							OK					

Flags

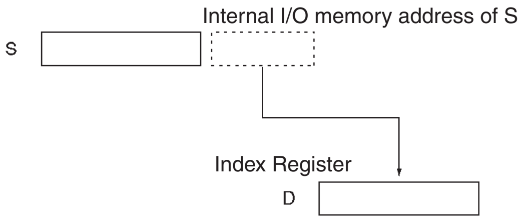
Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In CS1D CPU Units for Duplex Systems, CS1 CPU Units, and CJ1 CPU Units, these are turned OFF.

Function

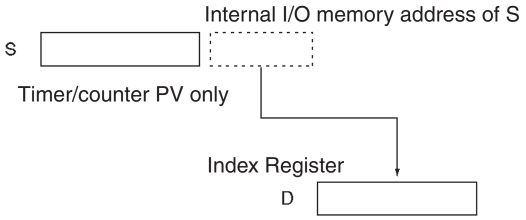
● MOVR

MOVR(560) finds the PLC memory address (absolute address) of S and writes that address in D (an Index Register).



● MOVRW

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).



Precautions

● MOVR

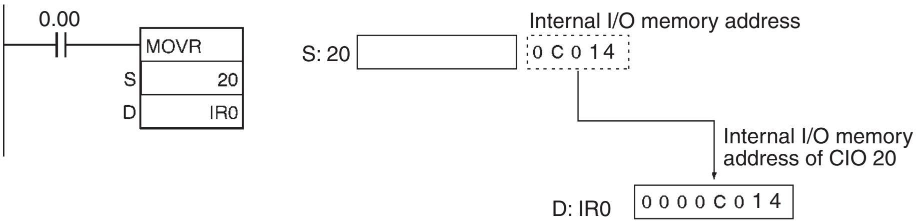
- The internal I/O memory address (excluding the timer/counter PV) is set in the index register (IR0 to 15) using this instruction.
- If S is specified using a regular I/O memory address (address based on area type), this will be automatically converted to an internal I/O memory address and stored in D.
- 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.
- 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.

● MOVRW

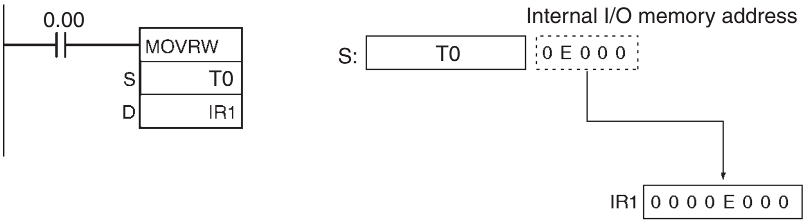
- MOVRW(561) will set the PLC memory address of the timer or counter's PV in D.
- Use MOVRW(561) to write the PLC memory address of the timer/counter PV in D.
- MOVRW(561) cannot set the PLC memory addresses of data area words, bits, or timer/counter Completion Flags. Use MOVR(560) to set these PLC memory addresses.

Example Programming

When CIO 0.00 is ON in the following example, MOVR(560) writes the PLC memory address of CIO 20 to IR0.



When CIO 0.00 is ON in the following example, MOVRW(561) writes the PLC memory address for the PV of timer T0 to IR1.

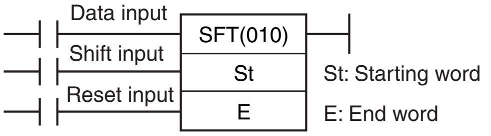


Refer to the CS/CJ Series Operation Manual or the *CJ2 CPU Unit Software Operation Manual* (W473) for specific PLC memory addresses.

Data Shift Instruction

SFT

Instruction	Mnemonic	Variations	Function code	Function
SHIFT REGISTER	SFT	---	010	Operates a shift register.

	SFT
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

* OK as long as words in the CIO Area, Work Area, or Holding Area are specified in the function block instance allocation areas.

Operands

Operand	Description	Data type	Size
St	Starting word	UINT	Variable
E	End word	UINT	Variable

● Operand Specifications

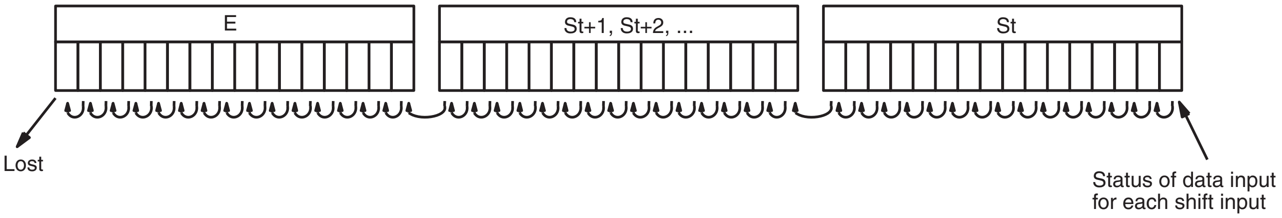
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
St E	OK	OK	OK	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	• ON if the indirect IR address for St and E is not in the CIO, WR or HR data areas. • OFF in all other cases.

Function

- 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.



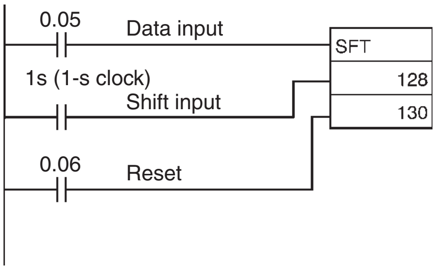
Precautions

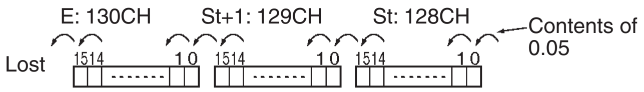
- Do not use more than one SFT(010) instructions with overlapping shift words. The results will not be dependable.
- St and E must be in the same data area.
- The bit data shifted out of the shift register is discarded.
- When the reset input turns ON, all bits in the shift register from the rightmost 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.

Example Programming

● Shift Register Exceeding 16 Bits

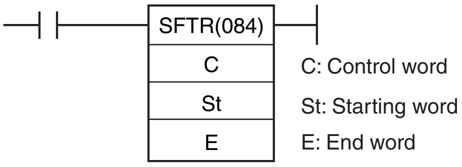
The following example shows a 48-bit shift register using words CIO 128 to CIO 130. A 1-s clock pulse is used so that the execution condition produced by CIO 0.05 is shifted into a 3-word register between CIO 128.00 and CIO 130.15 every second.





SFTR

Instruction	Mnemonic	Variations	Function code	Function
REVERSIBLE SHIFT REGISTER	SFTR	@SFTR	084	Creates a shift register that shifts data to either the right or the left.

	SFTR
Symbol	

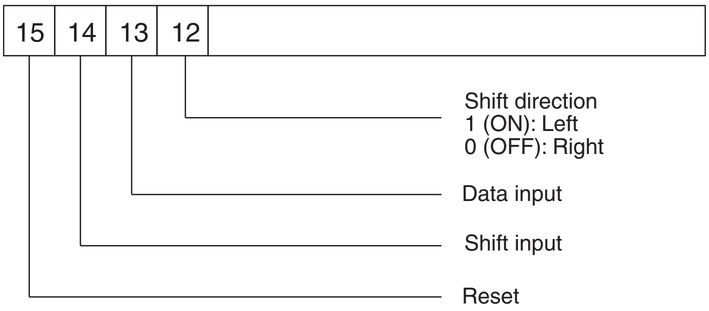
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	UINT	1
St	Starting word	UINT	Variable
E	End word	UINT	Variable

C: Control Word



Note St and E must be in the same data area.

● Operand Specifications

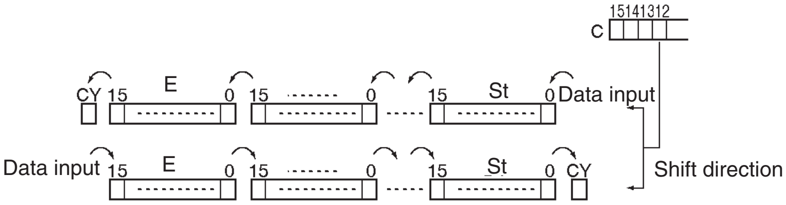
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
St												---						
E																		

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.

Function

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)

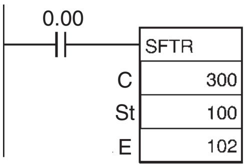


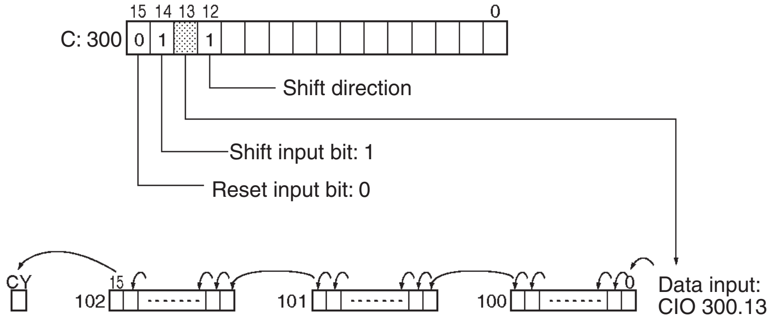
Note
- The above shift operations are applicable when the reset bit (bit 15 of C) is set to OFF.
- When reset (bit 15 of C) turns ON all bits in the shift register, from St to E will be reset (i.e., set to 0).

Example Programming

- Shifting Data

If shift input CIO 300.14 goes ON when CIO 0.00 is ON, and the reset bit CIO 300.15 is OFF, words CIO 100 through CIO 102 will shift one bit in the direction designated by CIO 300.12 (e.g., 1: Right) and the contents of input bit CIO 300.13 will be shifted into the rightmost bit, CIO 100.00. The contents of CIO 102.15 will be shifted to the Carry Flag (CY).



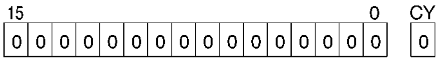


- Resetting Data

If CIO 300.14 is ON when CIO 0.00 is ON, and the reset bit, CIO 300.15, is ON, words CIO 100 through CIO 102 and the Carry Flag will be reset to OFF.

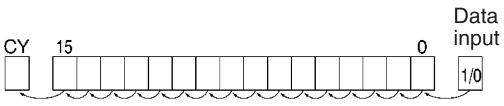
● Controlling Data

Resetting Data



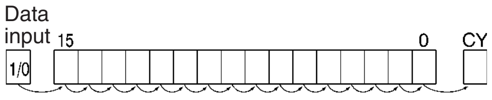
All bits from St to E and the Carry Flag are set to 0 and no other data can be received when the reset input bit (bit 15 of C) is ON.

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.

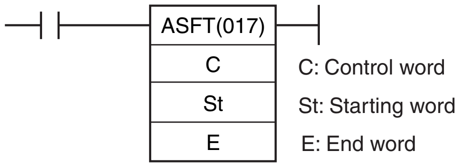
Shifting Data Right (from Leftmost to Rightmost Bit



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.

ASFT

Instruction	Mnemonic	Variations	Function code	Function
ASYNCHRONOUS SHIFT REGISTER	ASFT	@ASFT	017	Shifts all non-zero word data within the specified word range either towards St or toward E, replacing 0000Hex word data.

Symbol	ASFT
	

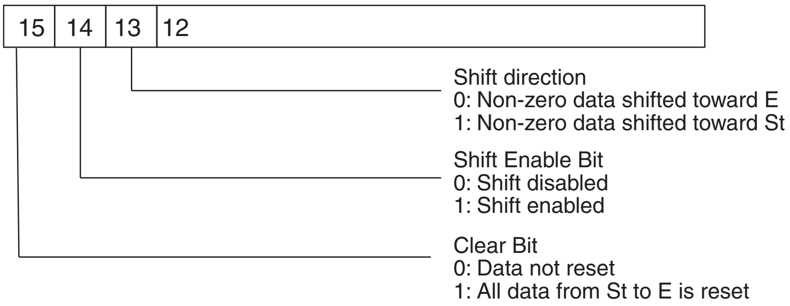
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	UINT	1
St	Starting word	UINT	Variable
E	End word	UINT	Variable

C: Control Word



Note St and E must be in the same data area.

● Operand Specifications

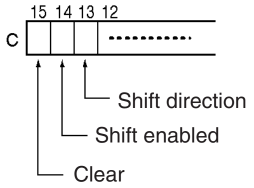
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
St												---						
E																		

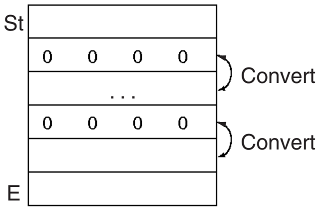
Flags

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.

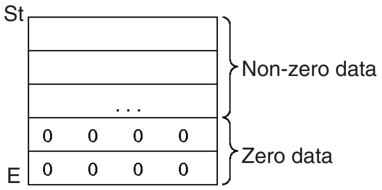
Function

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.





Note 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).

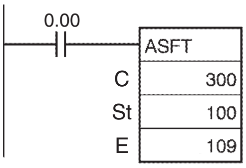


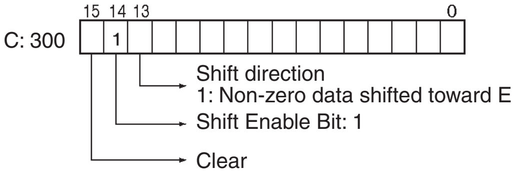
ASFT(017) can be processed in the background. Refer to the SYSMAC CS/CJ/NSJ Series PLC Programming Manual (W394) or the CJ2 CPU Unit Software Operation Manual (W473) for details.

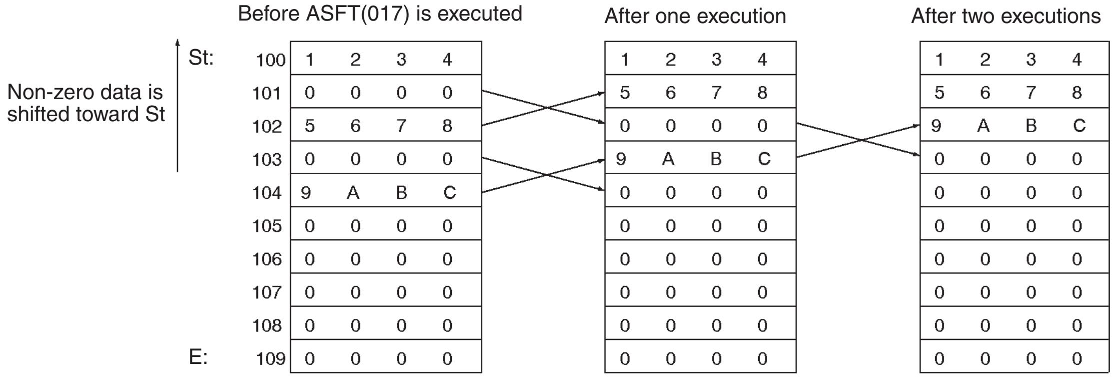
Example Programming

- Shifting Data

  If the Shift Enable Bit, CIO 300.14, goes ON when CIO 0.00 is ON, all words with non-zero data content from CIO 100 through CIO 109 will be shifted in the direction designated by the Shift Direction Bit, CIO 300.13 (e.g., 1: Toward St) if the word to the left of the non-zero data is all zeros.



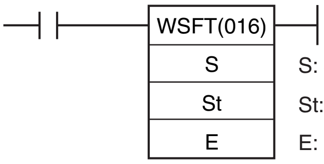




	Before ASFT(017) is executed	After one execution	After two executions
St: 100	1 2 3 4	1 2 3 4	1 2 3 4
101	0 0 0 0	5 6 7 8	5 6 7 8
102	5 6 7 8	0 0 0 0	9 A B C
103	0 0 0 0	9 A B C	0 0 0 0
104	9 A B C	0 0 0 0	0 0 0 0
105	0 0 0 0	0 0 0 0	0 0 0 0
106	0 0 0 0	0 0 0 0	0 0 0 0
107	0 0 0 0	0 0 0 0	0 0 0 0
108	0 0 0 0	0 0 0 0	0 0 0 0
E: 109	0 0 0 0	0 0 0 0	0 0 0 0

WSFT

Instruction	Mnemonic	Variations	Function code	Function
WORD SHIFT	WSFT	@WSFT	016	Shifts data between St and E in word units.

	WSFT
Symbol	 S: Source word St: Starting word E: End word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Control word	WORD	1
St	Starting word	UINT	Variable
E	End word	UINT	Variable

● Operand Specifications

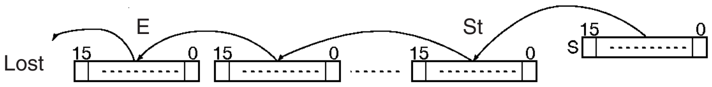
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
St											---	---						
E																		

Flags

Name	Label	Operation
Error Flag	ER	• ON when St is greater than E. • OFF in all other cases.

Function

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.

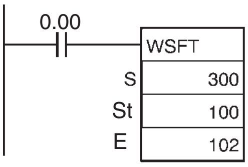


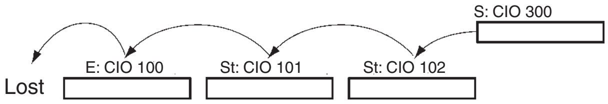
Precautions

- St and E must be in the same data area.
- 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.

Example Programming

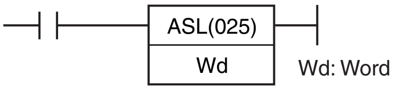
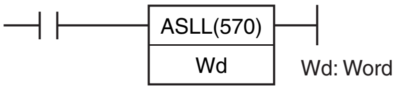
When CIO 0.00 is ON, data from CIO 100 through CIO 102 will be shifted one word toward E. The contents of CIO 300 will be stored in CIO 100 and the contents of CIO 102 will be lost.





ASL/ASLL

Instruction	Mnemonic	Variations	Function code	Function
ARITHMETIC SHIFT LEFT	ASL	@ASL	025	Shifts the contents of Wd one bit to the left.
DOUBLE SHIFT LEFT	ASLL	@ASLL	570	Shifts the contents of Wd and Wd +1 one bit to the left.

	ASL	ASLL
Symbol	 ASL(025) Wd Wd: Word	 ASLL(570) Wd Wd: Word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ASL	ASLL	ASL	ASLL
Wd	Word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ASL	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
ASLL	Wd												---						

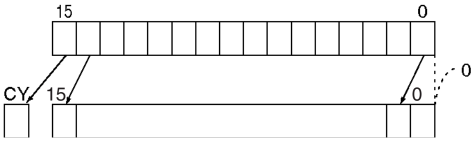
Flags

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.

Function

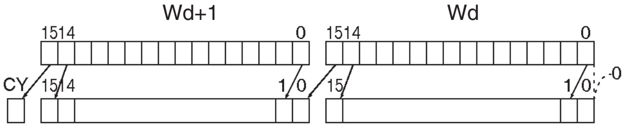
● ASL

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).



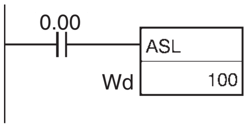
● ASLL

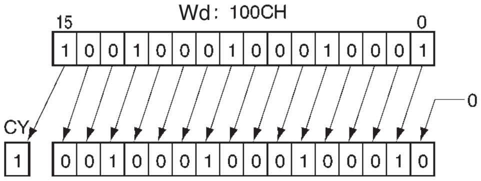
ASLL(570) shifts the contents of Wd and Wd +1 one bit to the left (from rightmost bit to leftmost bit). “0” is placed in the rightmost bit of Wd and the contents of the leftmost bit of Wd and Wd +1 are shifted into the Carry Flag (CY).



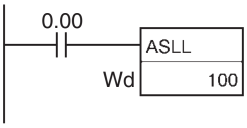
Example Programming

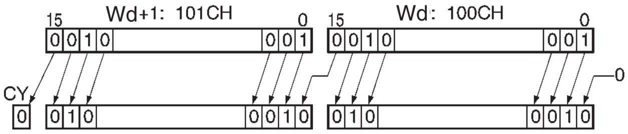
When CIO 0.00 is ON, CIO 100 will be shifted one bit to the left. “0” will be placed in CIO 100.00 and the contents of CIO 100.15 will be shifted to the Carry Flag (CY).





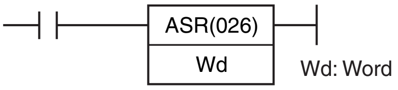
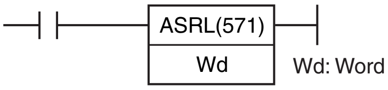
When CIO 0.00 is ON, word CIO 100 and CIO 101 will shift one bit to the left. “0” is placed into CIO 100.00 and the contents of CIO 101.15 will be shifted to the Carry Flag (CY).





ASR/ASRL

Instruction	Mnemonic	Variations	Function code	Function
ARITHMETIC SHIFT RIGHT	ASR	@ASL	026	Shifts the contents of Wd one bit to the right.
DOUBLE SHIFT RIGHT	ASRL	@ASLL	571	Shifts the contents of Wd and Wd +1 one bit to the right.

	ASR	ASRL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ASR	ASRL	ASR	ASRL
Wd	Word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ASR	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
ASRL	Wd												---						

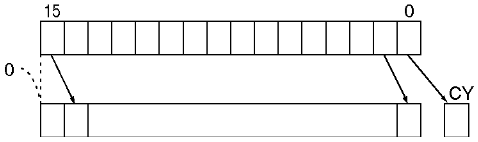
Flags

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

Function

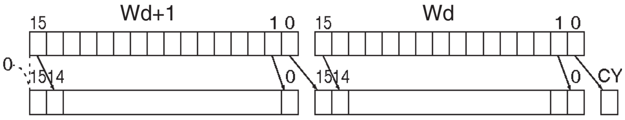
● ASR

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).



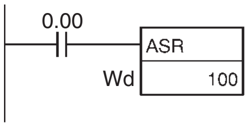
● ASRL

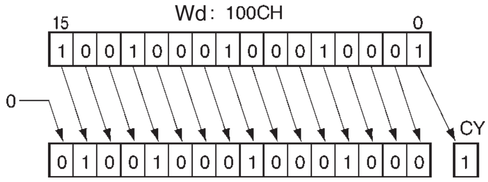
ASRL(571) shifts the contents of Wd and Wd +1 one bit to the right (from leftmost bit to rightmost bit). "0" will be placed in the leftmost bit of Wd +1 and the contents of the rightmost bit of Wd will be shifted into the Carry Flag (CY).



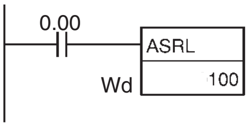
Example Programming

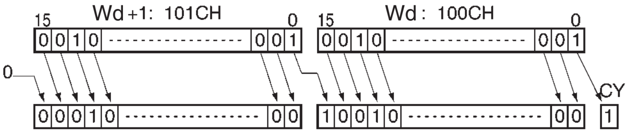
When CIO 0.00 is ON, word CIO 100 will shift one bit to the right. "0" will be placed in CIO 100.15 and the contents of CIO 100.00 will be shifted to the Carry Flag (CY).





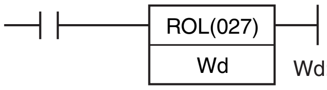
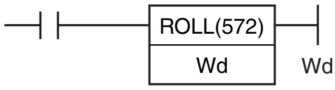
When CIO 0.00 is ON, word CIO 100 and CIO 101 will shift one bit to the right. "0" will be placed into CIO 100.15 and the contents of CIO 100.00 will be shifted to the Carry Flag (CY).





ROL/ROLL

Instruction	Mnemonic	Variations	Function code	Function
ROTATE LEFT	ROL	@ROL	027	Shifts all Wd bits one bit to the left including the Carry Flag (CY).
DOUBLE ROTATE LEFT	ROLL	@ROLL	572	Shifts all Wd and Wd +1 bits one bit to the left including the Carry Flag (CY).

	ROL	ROLL
Symbol	 Wd: Word	 Wd: Word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ROL	ROLL	ROL	ROLL
Wd	Word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ROL	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
ROLL	Wd												---						

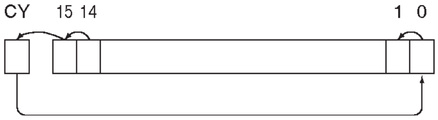
Flags

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.

Function

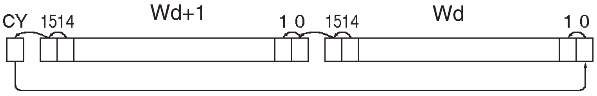
● ROL

ROL(027) shifts all bits of Wd including the Carry Flag (CY) to the left (from rightmost bit to leftmost bit).



● ROLL

ROLL(572) shifts all bits of Wd and Wd +1 including the Carry Flag (CY) to the left (from rightmost bit to leftmost bit).

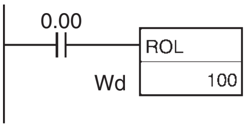


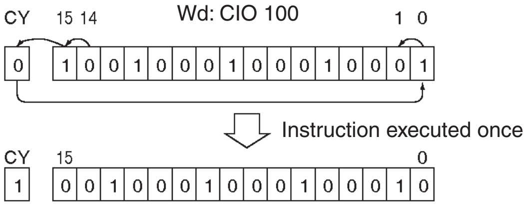
Hint

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.

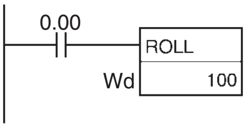
Example Programming

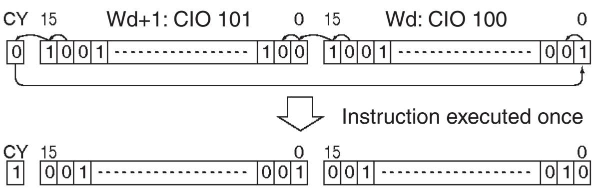
When CIO 0.00 is ON, word CIO 100 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 100.15 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 100.00.





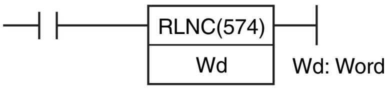
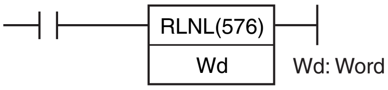
When CIO 0.00 is ON, word CIO 100, CIO 101 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 100.15 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 100.00.





RLNC/RLNL

Instruction	Mnemonic	Variations	Function code	Function
ROTATE LEFT WITHOUT CARRY	RLNC	@RLNC	574	Shifts all Wd bits one bit to the left not including the Carry Flag (CY).
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	@RLNL	576	Shifts all Wd and Wd +1 bits one bit to the left not including the Carry Flag (CY).

	RLNC	RLNL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		RLNC	RLNL	RLNC	RLNL
Wd	Word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
RLNC	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
RLNL	Wd												---						

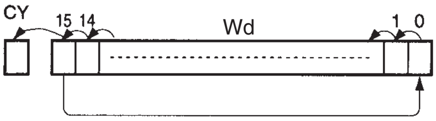
Flags

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.

Function

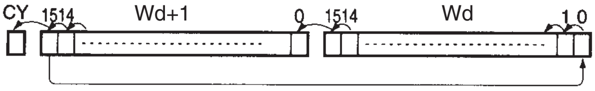
● RLNC

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).



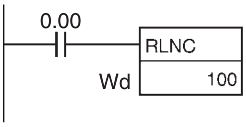
● RLNL

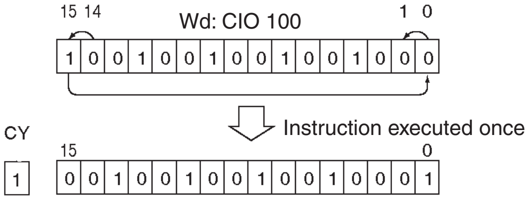
RLNL(576) shifts all bits of Wd and Wd +1 to the left (from rightmost bit to leftmost bit). The contents of the leftmost bit of Wd +1 is shifted to the rightmost bit of Wd, and to the Carry Flag (CY).



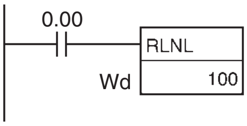
Example Programming

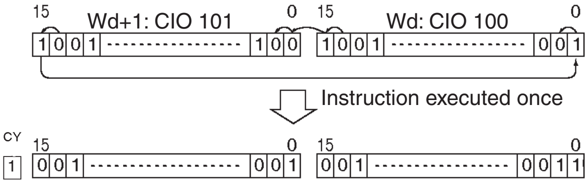
When CIO 0.00 is ON, word CIO 100 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 100.15 will be shifted to CIO 100.00.





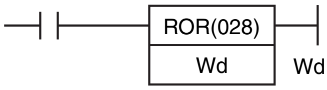
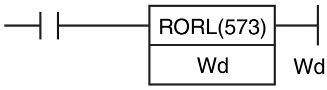
When CIO 0.00 is ON, word CIO 100 and CIO 101 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 101.15 will be shifted to CIO 100.00.





ROR/RORL

Instruction	Mnemonic	Variations	Function code	Function
ROTATE RIGHT	ROR	@ROR	028	Shifts all Wd bits one bit to the right including the Carry Flag (CY).).
DOUBLE ROTATE RIGHT	RORL	@RORL	573	Shifts all Wd and Wd +1 bits one bit to the right including the Carry Flag (CY).

	ROR	RORL
Symbol	 Wd: Word	 Wd: Word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ROR	RORL	ROR	RORL
Wd	Word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ROR	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
RORL	Wd												---						

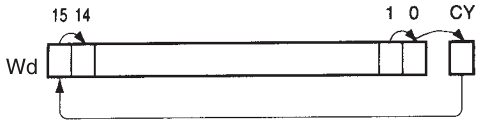
Flags

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.

Function

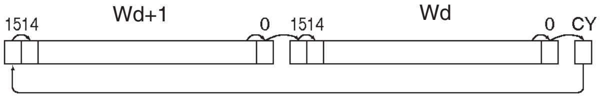
● ROR

ROR(028) shifts all bits of Wd including the Carry Flag (CY) to the right (from leftmost bit to rightmost bit).



● RORL

RORL(573) shifts all bits of Wd and Wd +1 including the Carry Flag (CY) to the right (from leftmost bit to rightmost bit).

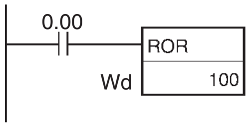


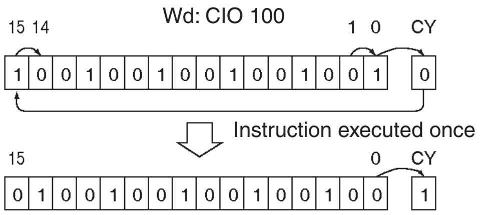
Hint

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.

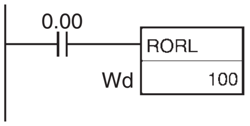
Example Programming

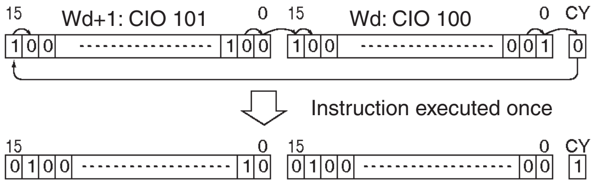
When CIO 0.00 is ON, word CIO 100 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 100.00 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 100.15.





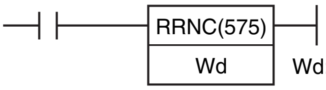
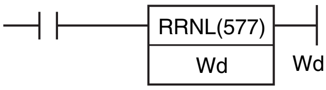
When CIO 0.00 is ON, word CIO 100, CIO 101 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 101.00 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 101.15.





RRNC/RRNL

Instruction	Mnemonic	Variations	Function code	Function
ROTATE RIGHT WITHOUT CARRY	RRNC	@RRNC	575	Shifts all Wd bits one bit to the right not including the Carry Flag (CY).
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	@RRNL	577	Shifts all Wd and Wd +1 bits one bit to the right not including the Carry Flag (CY).

	RRNC	RRNL
Symbol	 Wd: Word	 Wd: Word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		RRNC	RRNL	RRNC	RRNL
Wd	Word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
RRNC	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
RRNL	Wd												---						

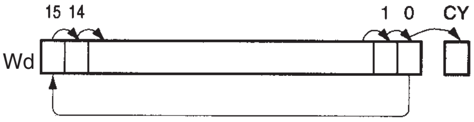
Flags

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.

Function

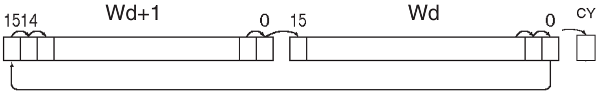
● RRNC

RRNC(575) shifts all bits of Wd to the right (from leftmost bit to rightmost bit) 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).



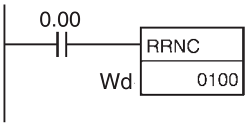
● RRNL

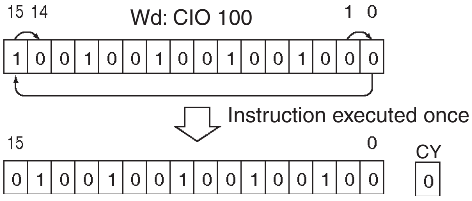
RRNL(577) shifts all bits of Wd and Wd +1 to the right (from leftmost bit to rightmost bit) not including the Carry Flag (CY). The contents of the rightmost bit of Wd +1 is shifted to the leftmost bit of Wd, and to the Carry Flag (CY).



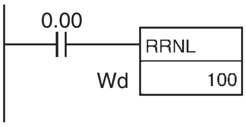
Example Programming

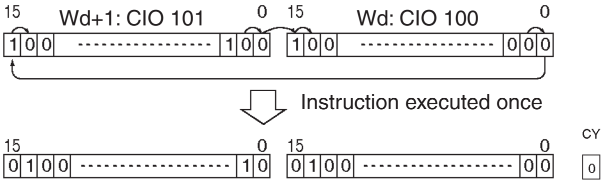
When CIO 0.00 is ON, word CIO 100 will shift one bit to the right (excluding the Carry Flag (CY)). The contents of CIO 100.00 will be shifted to CIO 100.15.





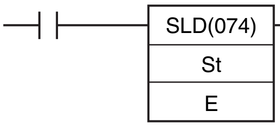
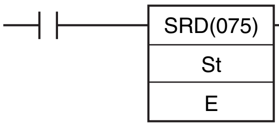
When CIO 0.00 is ON, words CIO 100 and CIO 101 will shift one bit to the right, (excluding the Carry Flag (CY)). The contents of CIO 100.00 will be shifted to CIO 101.15.





SLD/SRD

Instruction	Mnemonic	Variations	Function code	Function
ONE DIGIT SHIFT LEFT	SLD	@SLD	074	Shifts data by one digit (4 bits) to the left.
ONE DIGIT SHIFT RIGHT	SRD	@SRD	075	Shifts data by one digit (4 bits) to the right.

	SLD	SRD
Symbol	 St: Starting Word E: End word	 St: Starting Word E: End word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
St	Starting Word	UINT	Variable
E	End Word	UINT	Variable

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
St E	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	• ON when St is greater than E. • OFF in all other cases.

Function

● SLD

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.

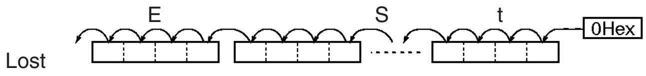
● SRD

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.

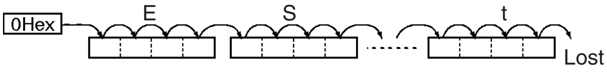
Precautions

- St and E must be in the same data area.
- 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) and SRD(075) is being executed, causing the shift operation to stop halfway through.

■SLD



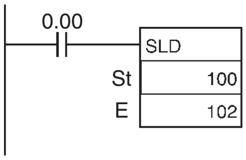
■SRD

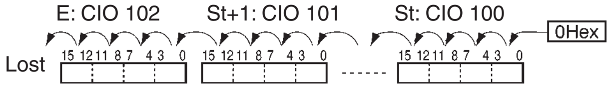


Example Programming

● SLD

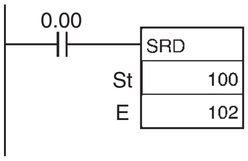
When CIO 0.00 is ON, words CIO 100 through CIO 102 will shift by one digit (4 bits) to the left. A zero will be placed in bits 0 to 3 of word CIO 100 and the contents of bits 12 to 15 of CIO 102 will be lost.

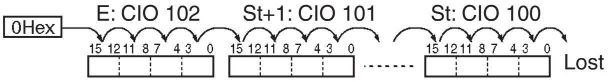




● SRD

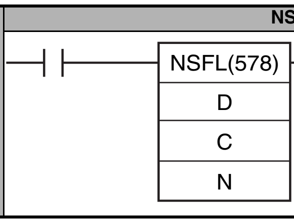
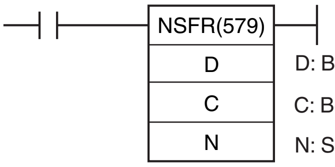
When CIO 0.00 is ON, words CIO 100 through CIO 102 will shift by one digit (4 bits) to the right. A zero will be placed in bits 12 to 15 of CIO 102 and the contents of bits 0 to 3 of word CIO 100 will be lost.





NSFL/NSFR

Instruction	Mnemonic	Variations	Function code	Function
SHIFT N-BIT DATA LEFT	NSFL	@NSFL	578	Shifts the specified number of bits to the left.
SHIFT N-BIT DATA RIGHT	NSFR	@NSFR	579	Shifts the specified number of bits to the right.

	NSFL	NSFR
Symbol	 D: Beginning word for shift C: Beginning bit N: Shift data length	 D: Beginning word for shift C: Beginning bit N: Shift data length

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	Beginning word for shift	UINT	Variable
C	Beginning bit	UINT	1
N	Shift data length	UINT	1

C: Beginning bit

0000 to 000F hex (0 to 15)

N: Shift data length

0000 to FFFF hex (0 to 65535)

Note All words in the shift register must be in the same area.

● Operand Specifications

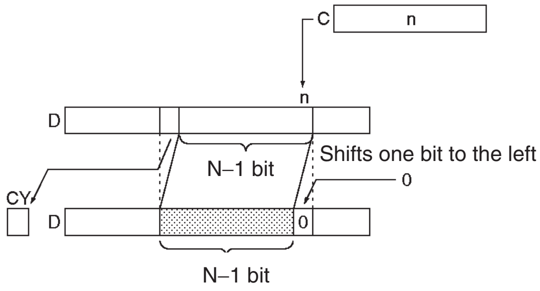
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK	OK						
N																		

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.

Function

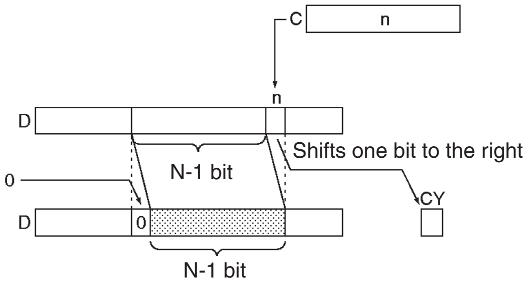
● NSFL



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).

Note
- 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.

● NSFR



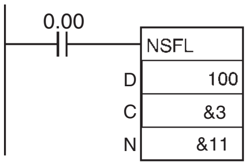
NSFR(579) shifts the specified number of bits by the shift data length (N) from the beginning bit (C) in the rightmost word as designated by D one bit to the right (towards the rightmost word and the rightmost bit). "0" will be placed into the beginning bit and the contents of the rightmost bit in the shift area will be shifted to the Carry Flag (CY).

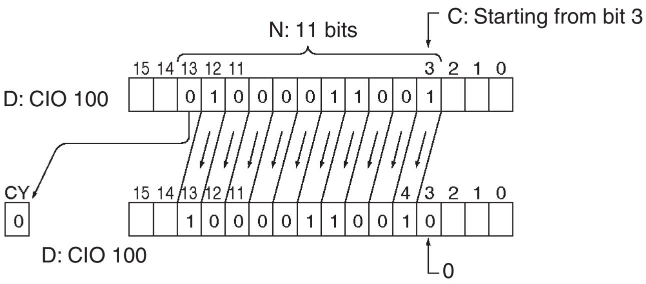
Note
- When the shift data length (N) is 0, the contents of the beginning bit will be copied to the Carry Flag (CY), and its contents will not be changed.
- Only the bits shifted into rightmost word in the shift area (i.e. leftmost word data) will be changed

Example Programming

● NSFL

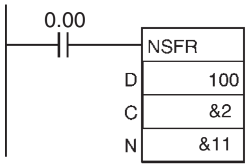
When CIO 0.00 is ON, all bits from the beginning bit 3 to the shift data length (B hex) will be shifted one bit to the left (from the rightmost bit to the leftmost bit). "0" will be placed into bit 3 of CIO 100. The contents of the leftmost bit in the shift area (bit 13 of CIO 100) are copied into the Carry Flag (CY).

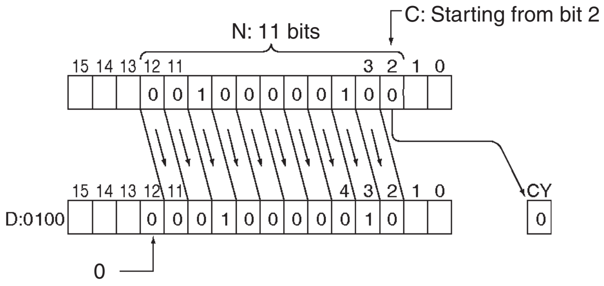




● NSFR

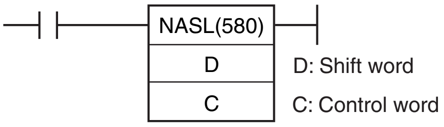
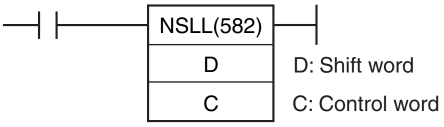
When CIO 0.00 is ON, all bits from the beginning bit 2 to end of the shift data length 11 bits (B hex), will be shifted one bit to the right, (from the leftmost bit to the rightmost bit). “0” is shifted into bit 12 of CIO 100. The contents of the rightmost bit in the shift area (bit 2 of CIO 100) are copied into the Carry Flag (CY).





NASL/NSLL

Instruction	Mnemonic	Variations	Function code	Function
SHIFT N-BITS LEFT	NASL	@NASL	580	Shifts the specified 16 bits of word data to the left by the specified number of bits.
DOUBLE SHIFT N-BITS LEFT	NSLL	@NSLL	582	Shifts the specified 32 bits of word data to the left by the specified number of bits.

	NASL	NSLL
Symbol		

Applicable Program Areas

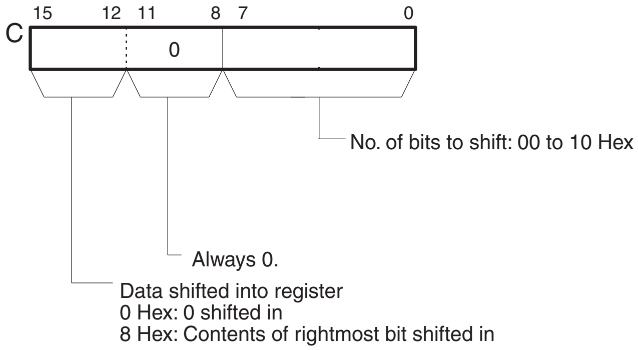
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

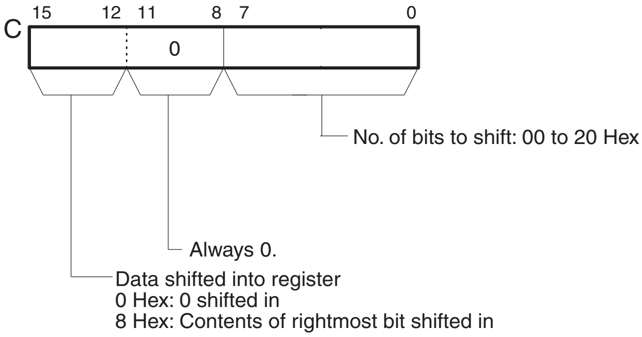
Operand	Description	Data type		Size	
		NASL	NSLL	NASL	NSLL
D	Shift Word	UINT	UDINT	1	2
C	Control word	UINT	UDINT	1	1

C: Control word

■ **NASL**



■ **NSLL**



● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
NASL	D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
	C											OK							
NSLL	D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
	C											OK	OK						

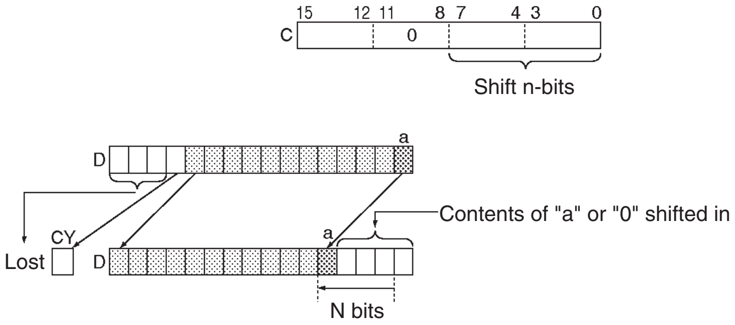
Flags

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.

Function

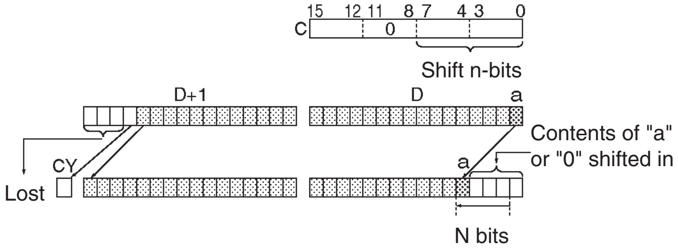
● NASL

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.



● NSLL

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.

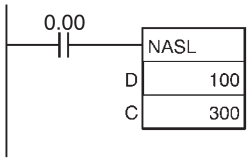


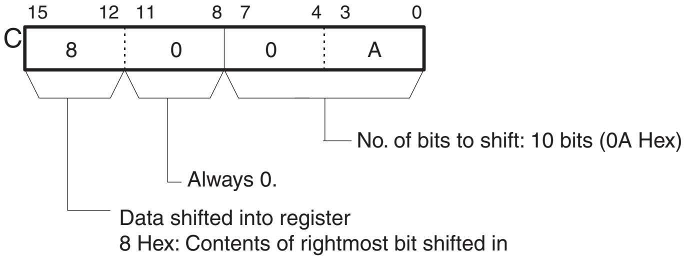
Precautions

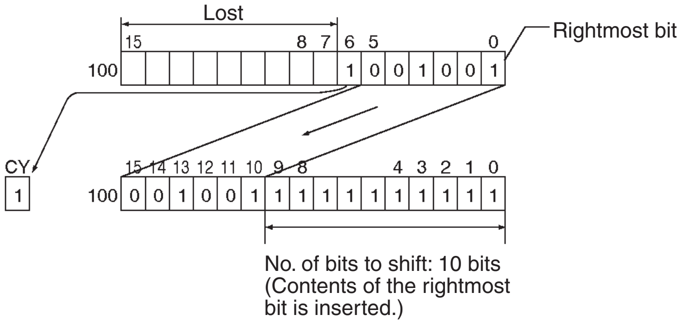
- For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.
- When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

Example Programming

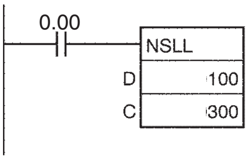
When CIO 0.00 is ON, The contents of CIO 100 is shifted 10 bits to the left (from the rightmost bit to the leftmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 300 (control data). The contents of bit 0 of CIO 100 is copied into bits from which data was shifted and the contents of the rightmost bit which was shifted out of range is shifted into the Carry Flag (CY). All other data is lost.

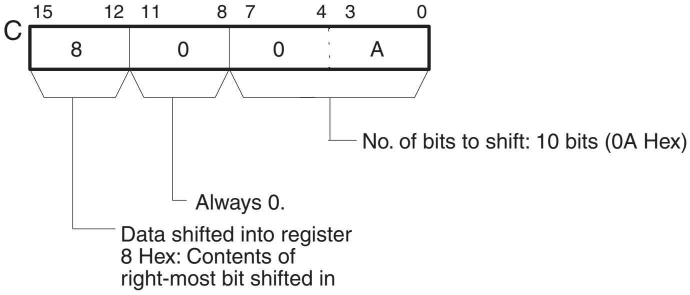


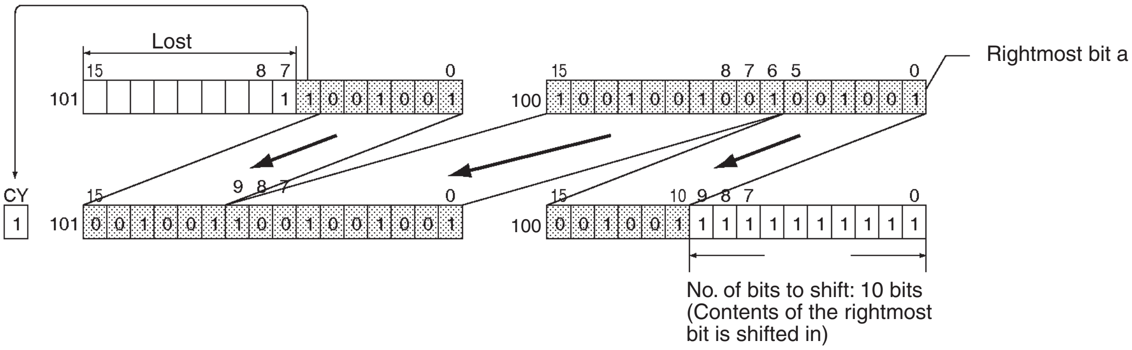




When CIO 0.00 is ON, CIO 100 and CIO 101 will be shifted to the left (from the rightmost bit to the leftmost bit) by 10 bits. The number of bits to shift is specified in bits 0 to 7 of word CIO 300 (control data). The contents of bit 0 of CIO 100 is copied into bits from which data was shifted and the contents of the rightmost bit which was shifted out of range is shifted into the Carry Flag (CY). All other data is lost.

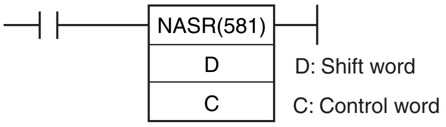
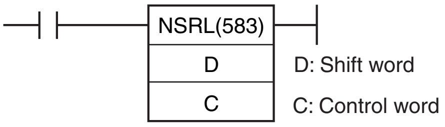






NASR/NSRL

Instruction	Mnemonic	Variations	Function code	Function
SHIFT N-BITS RIGHT	NASR	@NASR	581	Shifts the specified 16 bits of word data to the right by the specified number of bits.
DOUBLE SHIFT N-BITS RIGHT	NSRL	@NSRL	583	Shifts the specified 32 bits of word data to the right by the specified number of bits.

	NASR	NSRL
Symbol		

Applicable Program Areas

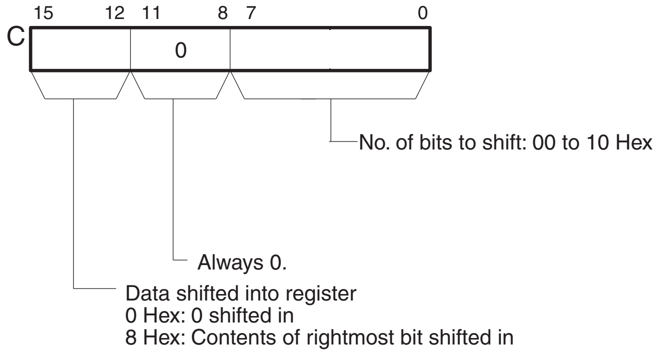
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

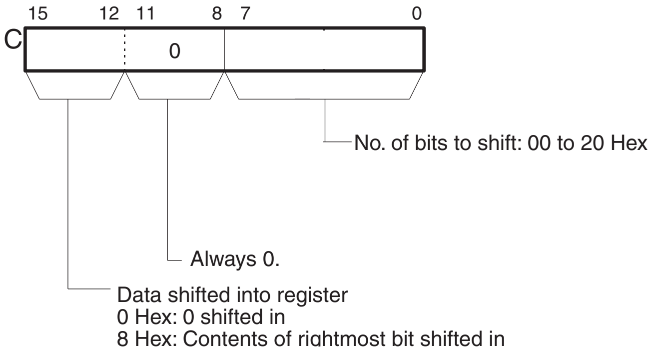
Operand	Description	Data type		Size	
		NASR	NSRL	NASR	NSRL
D	Shift Word	UINT	UDINT	1	2
C	Control word	UINT	UDINT	1	1

C: Control word

■ NASR



■ NSRL



● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
NASR	D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
	C											OK							
NSRL	D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
	C											OK	OK						

Flags

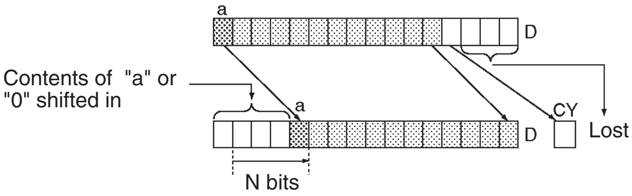
Name	Label	Operation
Error Flag	ER	• ON when the control word C (the number of bits to shift) is not within range. • OFF in all other cases.
Equals Flag	=	• ON when the shift result is 0. • OFF in all other cases.

Name	Label	Operation
Carry Flag	CY	• ON when 1 is shifted into the Carry Flag (CY). • OFF in all other cases.
Negative Flag	N	• ON when the leftmost bit is 1 as a result of the shift. • OFF in all other cases.

Function

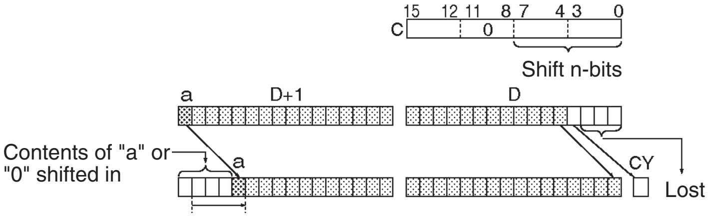
● NASR

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.



● NSRL

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.

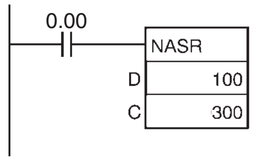


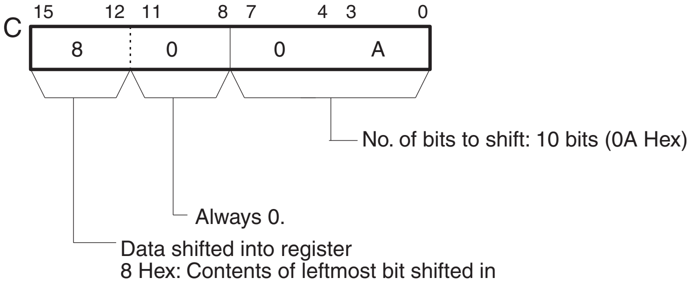
Precautions

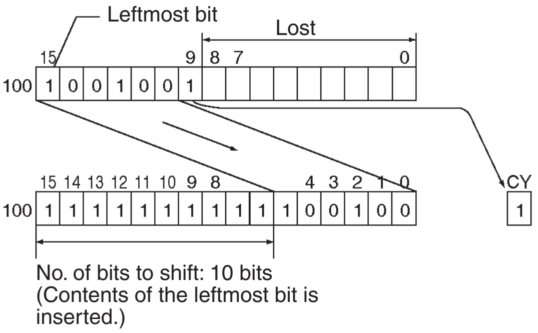
- For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is discarded.
- When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

Example Programming

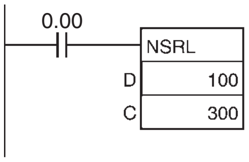
- When CIO 0.00 is ON, CIO 100 will be shifted 10 bits to the right (from the leftmost bit to the rightmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 300. The contents of bit 15 of CIO 100 is copied into the bits from which data was shifted and the contents of the leftmost bit of data which was shifted out of range, is shifted into the Carry Flag (CY). All other data is lost.

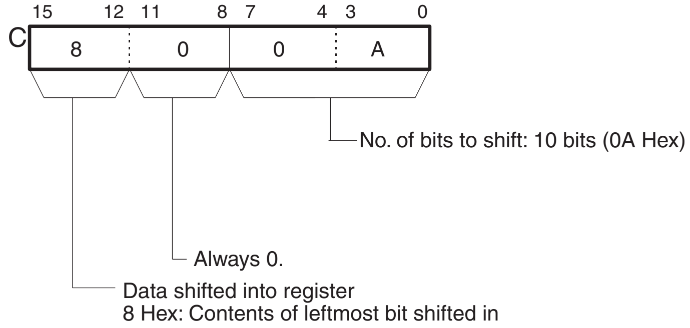


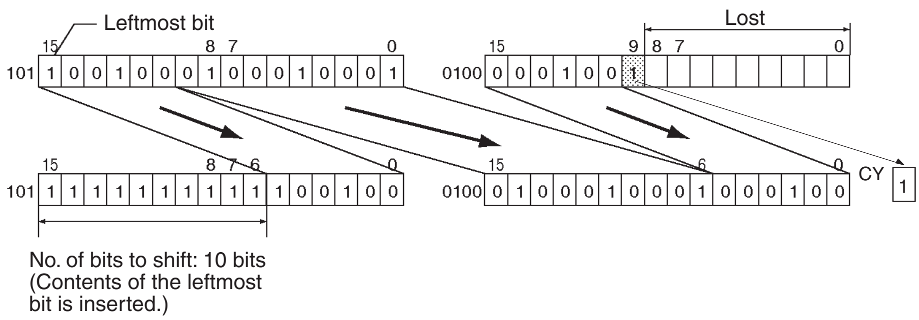




When CIO 0.00 is ON, CIO 100 and CIO 101 will be shifted 10 bits to the right (from the leftmost bit to the rightmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 300 (control data). The contents of bit 15 of CIO will be copied into the bits from which data was shifted and the contents of the leftmost bit of data which was shifted out of range will be shifted into the Carry Flag (CY). All other data is lost.



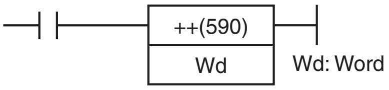
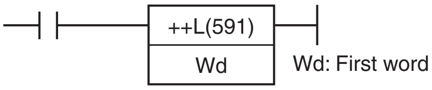




Increment/Decrement Instructions

++/++L

Instruction	Mnemonic	Variations	Function code	Function
INCREMENT BINARY	++	@++	590	Increments the 4-digit hexadecimal content of the specified word by 1.
DOUBLE INCREMENT BINARY	++L	@++L	591	Increments the 8-digit hexadecimal content of the specified words by 1.

	++	++L
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		++	++L	++	++L
Wd	++: Word ++L: First word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
++	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
++L	Wd												---	OK					

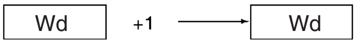
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	• ON if the result is 0000/0000 0000 after execution. • OFF in all other cases.
Carry Flag	CY	• ON if a digit in Wd/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/Wd+1 is ON after execution. • OFF in all other cases.

Function

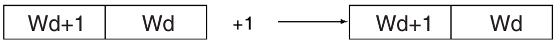
● ++

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.



● ++L

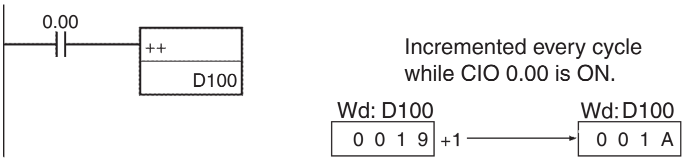
The ++L(591) instruction adds 1 to the 8-digit hexadecimal content of Wd+1 and Wd. The content of the specified words will be incremented by 1 every cycle as long as the execution condition of ++L(591) is ON. When the up-differentiated variation of this instruction (@++L(591)) is used, the content of the specified words is incremented only when the execution condition has gone from OFF to ON.



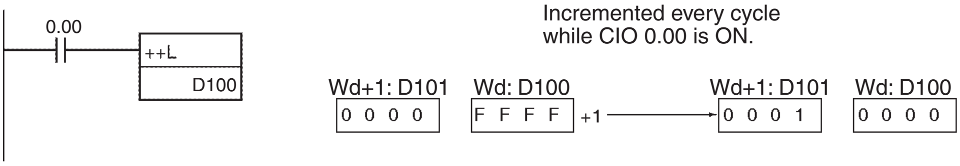
Example Programming

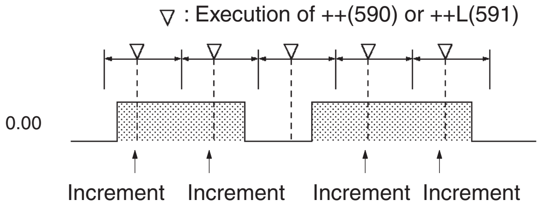
● Operation of ++(590)/++L(591)

In the following example, the content of D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.



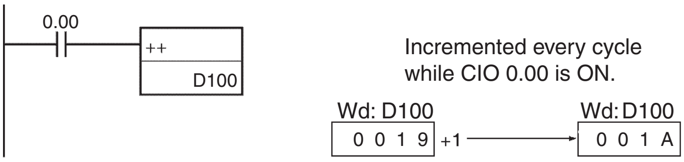
In the following example, the content of D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.



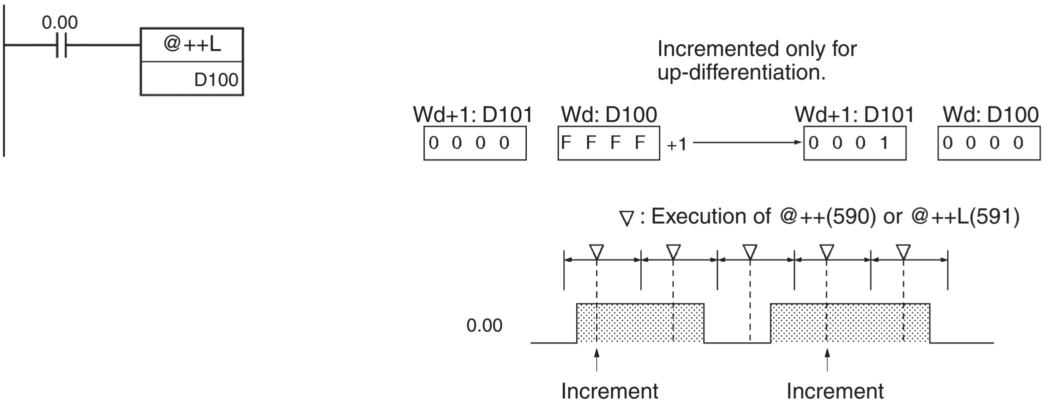


Operation of @++(590)/@++L(591)

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.

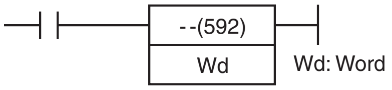
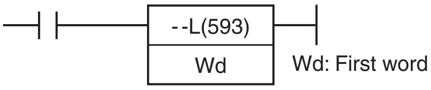


The up-differentiated variation is used in the following example, so the content of D101 and D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.



--/--L

Instruction	Mnemonic	Variations	Function code	Function
DECREMENT BINARY	--	@--	592	Decrements the 4-digit hexadecimal content of the specified word by 1.
DOUBLE DECREMENT BINARY	--L	@--L	593	Decrements the 8-digit hexadecimal content of the specified words by 1.

	--	--L
Symbol	 --(592) Wd Wd: Word	 --L(593) Wd Wd: First word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		--	--L	--	--L
Wd	--: Word --L: First word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
--	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
--L	Wd												---	OK					

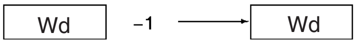
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	• ON if the result is 0000/0000 0000 after execution. • OFF in all other cases.
Carry Flag	CY	• ON if a digit in Wd/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/Wd+1 is ON after execution. • OFF in all other cases.

Function

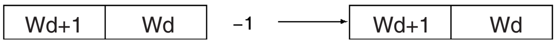
● --

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.



● --L

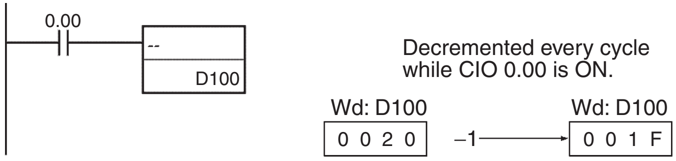
The --L(593) instruction subtracts 1 from the 8-digit hexadecimal content of Wd+1 and Wd. The content of the specified words will be decremented by 1 every cycle as long as the execution condition of --L(593) is ON. When the up-differentiated variation of this instruction (@--L(593)) is used, the content of the specified words is decremented only when the execution condition has gone from OFF to ON.



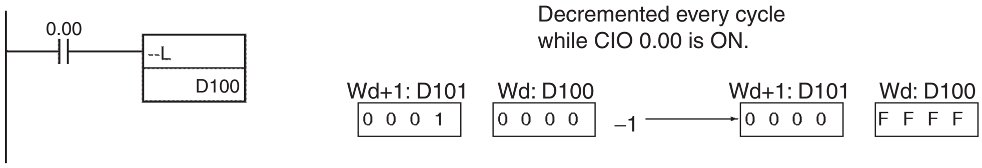
Example Programming

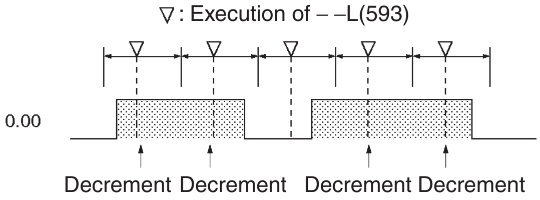
● Operation of --(592)/--L(593)

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.



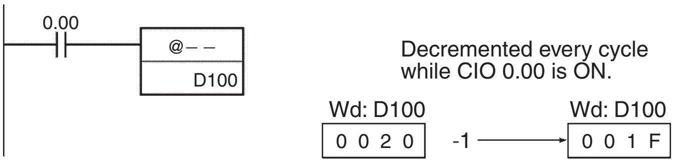
In the following example, the 8-digit hexadecimal content of D101 and D100 will be decremented by 1 every cycle as long as CIO 0.00 is ON.



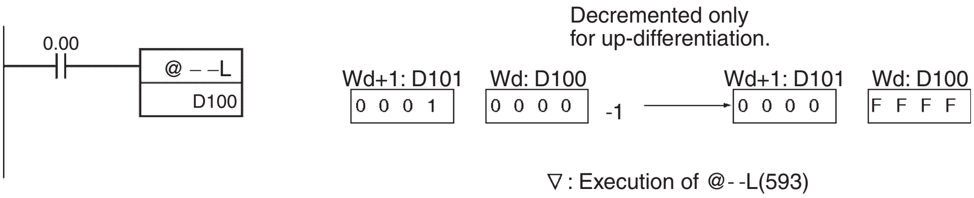


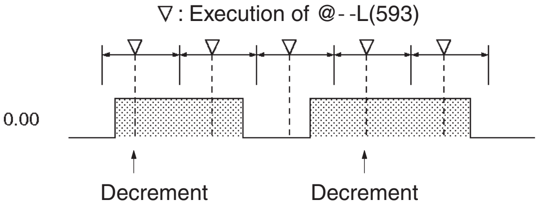
● Operation of @--(592)/@--L(593)

In the following example, the content of D100 will be decremented by 1 every cycle as long as CIO 0.00 is ON.



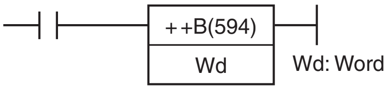
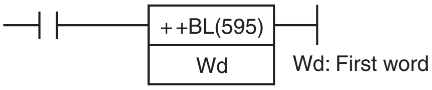
The up-differentiated variation is used in the following example, so the content of D101 and D100 will be decremented by 1 only when CIO 0.00 has gone from OFF to ON.





++B/++BL

Instruction	Mnemonic	Variations	Function code	Function
INCREMENT BCD	++B	@++B	594	Increments the 4-digit BCD content of the specified word by 1.
DOUBLE INCREMENT BCD	++BL	@++BL	595	Increments the 8-digit BCD content of the specified words by 1.

Symbol	++B	++BL
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		++	++L	++	++L
Wd	++B: Word ++BL: First word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
++B	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
++BL	Wd												---						

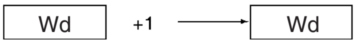
Flags

Name	Label	Operation
Error Flag	ER	• ON if the content of Wd/Wd+1 and Wd is not BCD. • OFF in all other cases.
Equals Flag	=	• ON if the result is 0000/0000 0000 after execution. • OFF in all other cases.
Carry Flag	CY	• ON if a digit in Wd/Wd+1 or Wd went from 9 to 0 during execution. • OFF in all other cases.

Function

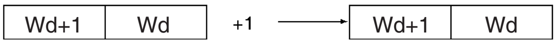
● ++B

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.



● ++BL

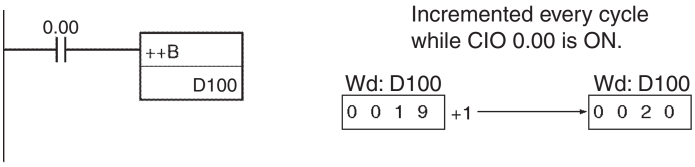
The ++BL(595) instruction adds 1 to the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be incremented by 1 every cycle as long as the execution condition of ++BL(595) is ON. When the up-differentiated variation of this instruction (@++BL(595)) is used, the content of the specified words is incremented only when the execution condition has gone from OFF to ON.



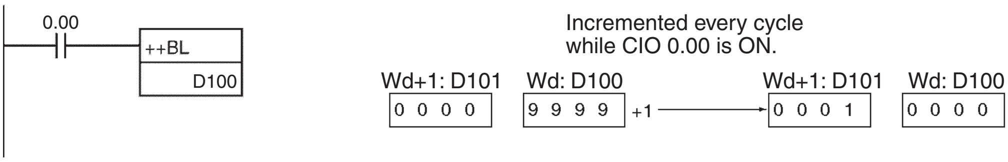
Example Programming

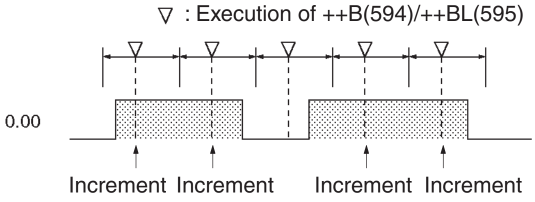
● Operation of ++B(594)/++BL(595)

In the following example, the BCD content of D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.



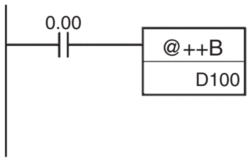
In the following example, the 8-digit BCD content of D101 and D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.

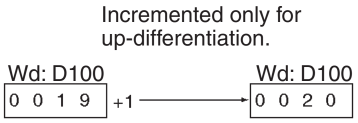




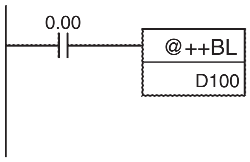
Operation of @++B(594)/@++BL(595)

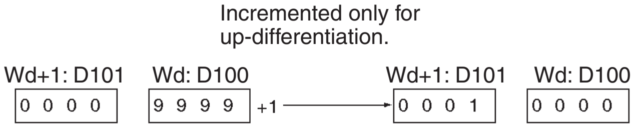
The up-differentiated variation is used in the following example, so the content of D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.

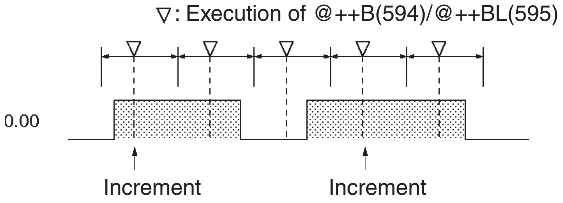




The up-differentiated variation is used in the following example, so the BCD content of D101 and D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.

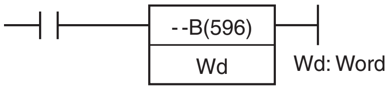
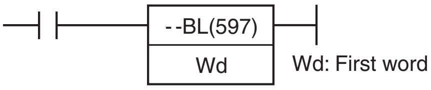






--B/--BL

Instruction	Mnemonic	Variations	Function code	Function
DECREMENT BCD	--B	@--B	596	Decrements the 4-digit BCD content of the specified word by 1.
DOUBLE DECREMENT BCD	--BL	@--BL	597	Decrements the 8-digit BCD content of the specified words by 1.

	--B	--BL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		--	--L	--	--L
Wd	--B: Word --BL: First word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
--B	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
--BL	Wd												---						

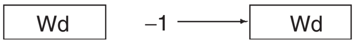
Flags

Name	Label	Operation
Error Flag	ER	• ON if the content of Wd/Wd+1 and Wd is not BCD. • OFF in all other cases.
Equals Flag	=	• ON if the result is 0000/0000 0000 after execution. • OFF in all other cases.
Carry Flag	CY	• ON if a digit in Wd/Wd+1 or Wd went from 0 to 9 during execution. • OFF in all other cases.

Function

● --B

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.



● --BL

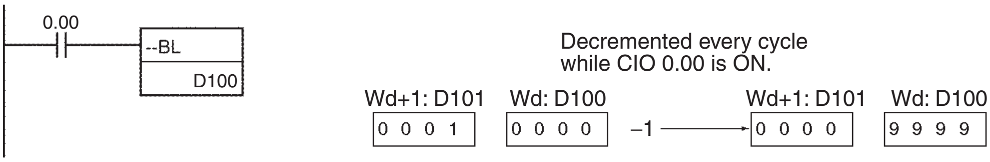
The --BL(597) instruction subtracts 1 from the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be decremented by 1 every cycle as long as the execution condition of --BL(597) is ON. When the up-differentiated variation of this instruction (@--BL(597)) is used, the content of the specified words is decremented only when the execution condition has gone from OFF to ON.

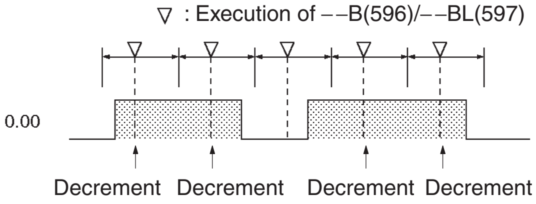
Example Programming

● Operation of --B(596)/--BL(597)

In the following example, the BCD content of D100 will be decremented by 1 every cycle as long as CIO 0.00 is ON.

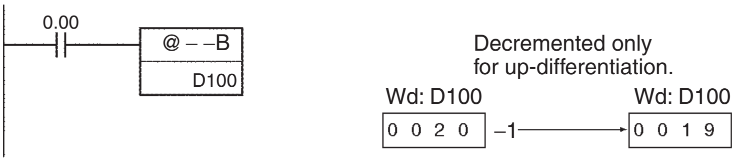
In the following example, the 8-digit BCD content of D101 and D100 will be decremented by 1 every cycle as long as CIO 0.00 is ON.



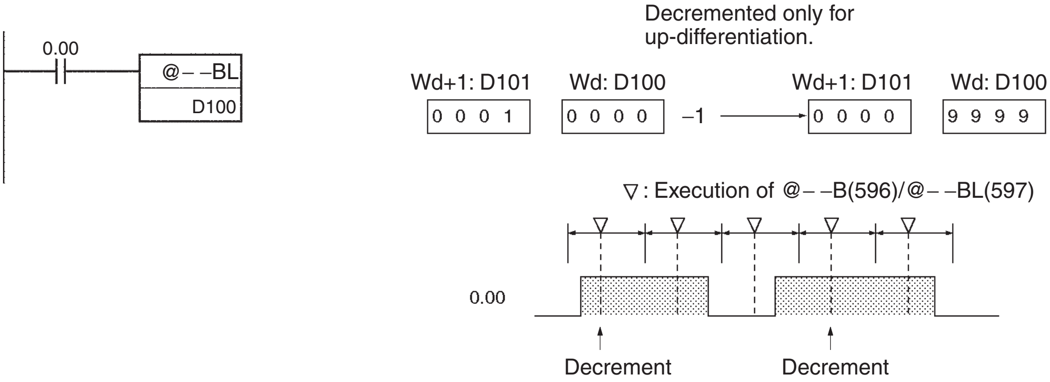


● Operation of @--B(596)/@--BL(597)

The up-differentiated variation is used in the following example, so the BCD content of D100 will be decremented by 1 only when CIO 0.00 has gone from OFF to ON.



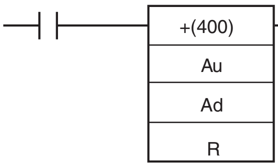
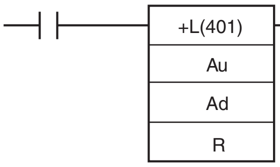
The up-differentiated variation is used in the following example, so the BCD content of D101 and D100 will be decremented by 1 only when CIO 0.00 has gone from OFF to ON.



Symbol Math Instructions

+/+L

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY ADD WITHOUT CARRY	+	@+	400	Adds 4-digit (single-word) hexadecimal data and/or constants.
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	@+L	401	Adds 8-digit (double-word) hexadecimal data and/or constants.

	+	+L
Symbol	 Au: Augend word Ad: Addend word R: Result word	 Au: 1st augend word Ad: 1st addend word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		+	+L	+	+L
Au	+: Augend word +L: First augend word	INT	DINT	1	2
Ad	+: Addend word +L: First addend word	INT	DINT	1	2
R	+: Result word +L: First result word	INT	DINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
+	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
+L	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	OK	---	---	---	---
	R											---							

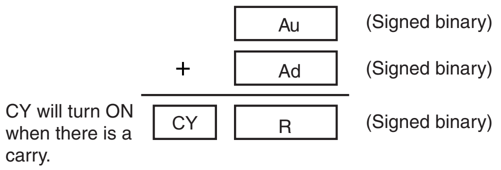
Flags

Name	Label	Operation	
		+	+L
Error Flag	ER	OFF	OFF
Equals Flag	=	• ON when the result is 0. • OFF in all other cases.	• 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.	• 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.	• 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 0000 to 7FFF hex. • OFF in all other cases.	• 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.	• ON when the leftmost bit of the result is 1. • OFF in all other cases.

Function

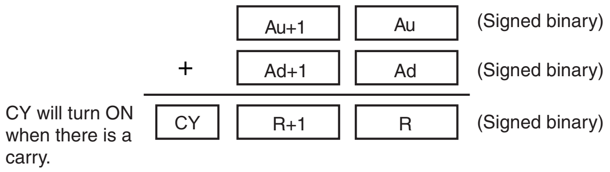
● +

+(400) adds the binary values in Au and Ad and outputs the result to R.

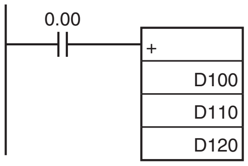


● +L

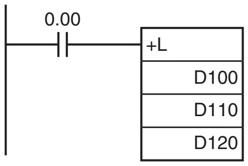
+L(401) adds the binary values in Au and Au+1 and Ad and Ad+1 and outputs the result to R.



Example Programming



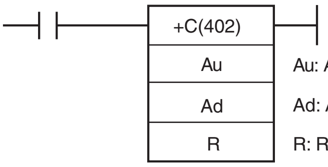
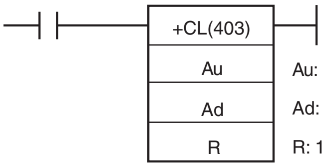
When CIO 0.00 is ON in this example, D100 and D110 will be added as 4-digit signed binary values and the result will be output to D120.



When CIO 0.00 is ON, D101 and D100 and D111 and D110 will be added as 8-digit signed binary values and the result will be output to D121 and D120.

+C/+CL

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY ADD WITH CARRY	+C	@+C	402	Adds 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	@+CL	403	Adds 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

	+C	+CL
Symbol	 Au: Augend word Ad: Addend word R: Result word	 Au: 1st augend word Ad: 1st addend word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		+C	+CL	+C	+CL
Au	+C: Augend word +CL: First augend word	INT	DINT	1	2
Ad	+C: Addend word +CL: First addend word	INT	DINT	1	2
R	+C: Result word +CL: First result word	INT	DINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
+C	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
+CL	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

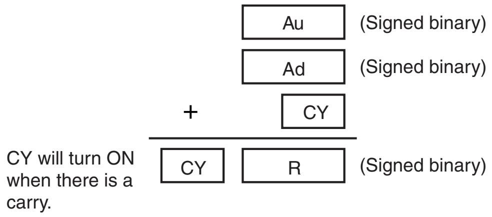
Flags

Name	Label	Operation	
		+C	+CL
Error Flag	ER	OFF	OFF
Equals Flag	=	• ON when the addition result is 0. • OFF in all other cases.	• 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.	• ON when the 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.	• 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 addition result of adding two negative numbers and CY is in the range 0000 to 7FFF hex. • OFF in all other cases.	• 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.	• ON when the leftmost bit of the result is 1. • OFF in all other cases.

Function

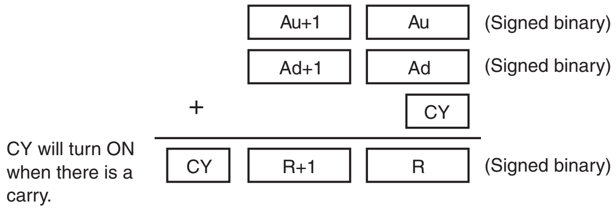
● +C

+C(402) adds the binary values in Au, Ad, and CY and outputs the result to R.



● +CL

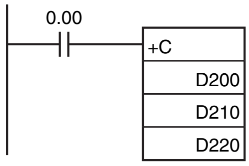
+CL(403) adds the binary values in Au and Au+1, Ad and Ad+1, and CY and outputs the result to R.



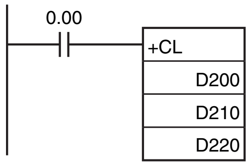
Hint

- To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Example Programming



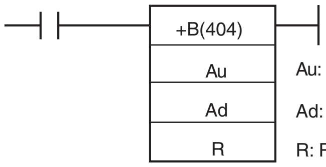
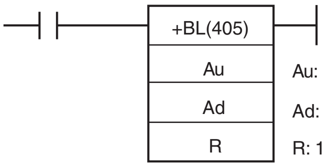
When CIO 0.00 is ON, D200, D210, and CY will be added as 4-digit signed binary values and the result will be output to D220.



When CIO 0.00 is ON, D201, D200, D211, D210, and CY will be added as 8-digit signed binary values, and the result will be output to D221 and D220.

+B/+BL

Instruction	Mnemonic	Variations	Function code	Function
BCD ADD WITHOUT CARRY	+B	@+B	404	Adds 4-digit (single-word) BCD data and/or constants.
DOUBLE BCD ADD WITHOUT CARRY	+BL	@+BL	405	Adds 8-digit (double-word) BCD data and/or constants.

	+B	+BL
Symbol	 Au: Augend word Ad: Addend word R: Result word	 Au: 1st augend word Ad: 1st addend word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		+B	+BL	+B	+BL
Au	+B: Augend word +BL: First augend word	WORD	DWORD	1	2
Ad	+B: Addend word +BL: First addend word	WORD	DWORD	1	2
R	+B: Result word +BL: First result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
+B	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
+BL	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

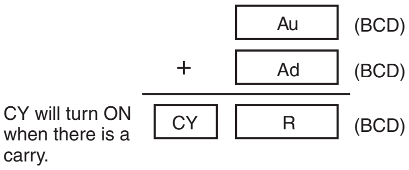
Flags

Name	Label	Operation	
		+B	+BL
Error Flag	ER	• ON when Au is not BCD. • ON when Ad is not BCD. • OFF in all other cases.	• 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.	• 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.	• ON when the addition results in a carry. • OFF in all other cases.

Function

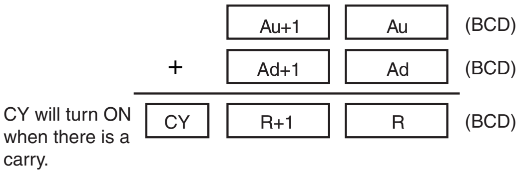
● +B

+B(404) adds the BCD values in Au and Ad and outputs the result to R.

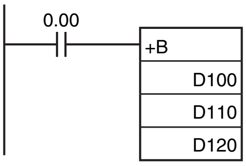


● +BL

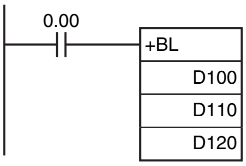
+BL(405) adds the BCD values in Au and Au+1 and Ad and Ad+1 and outputs the result to R, R+1.



Example Programming



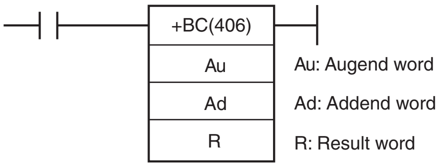
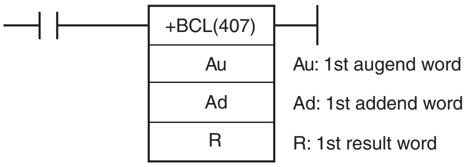
When CIO 0.00 is ON in the following example, D100 and D110 will be added as 4-digit BCD values, and the result will be output to D120.



When CIO 0.00 is ON in the following example, D101 and D100 and D111 and D110 will be added as 8-digit BCD values, and the result will be output to D121 and D120.

+BC/+BCL

Instruction	Mnemonic	Variations	Function code	Function
BCD ADD WITH CARRY	+BC	@+BC	406	Adds 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).
DOUBLE BCD ADD WITH CARRY	+BCL	@+BCL	407	Adds 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).

	+BC	+BCL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		+BC	+BCL	+BC	+BCL
Au	+BC: Augend word +BCL: First augend word	WORD	DWORD	1	2
Ad	+BC: Addend word +BCL: First addend word	WORD	DWORD	1	2
R	+BC: Result word +BCL: First result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
+BC	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
+BCL	Au, Ad	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

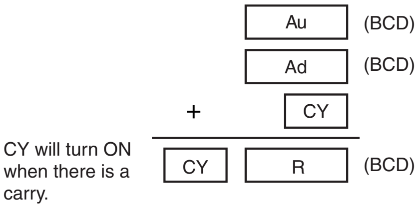
Flags

Name	Label	Operation	
		+BC	+BCL
Error Flag	ER	• ON when Au is not BCD. • ON when Ad is not BCD. • OFF in all other cases.	• 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.	• 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.	• ON when the addition results in a carry. • OFF in all other cases.

Function

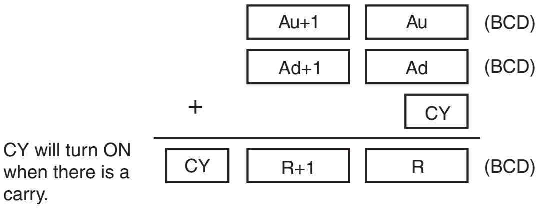
● +BC

+BC(406) adds BCD values in Au, Ad, and CY and outputs the result to R.



● +BCL

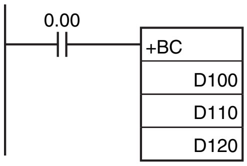
+BCL(407) adds the BCD values in Au and Au+1, Ad and Ad+1, and CY and outputs the result to R, R+1.



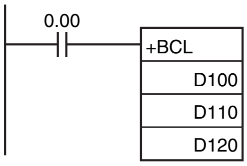
Hint

- To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Example Programming



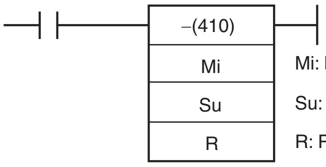
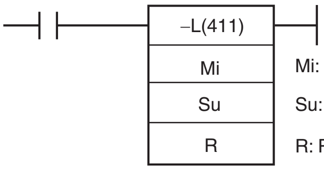
When CIO 0.00 is ON in the following example, D100, D110, and CY will be added as 4-digit BCD values, and the result will be output to D120.



When CIO 0.00 is ON in the following example, D101, D100, D111, D110, and CY will be added as 8-digit BCD values, and the result will be output to D121 and D120.

–/–L

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	@–	410	Subtracts 4-digit (single-word) hexadecimal data and/or constants.
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	@–L	411	Subtracts 8-digit (double-word) hexadecimal data and/or constants.

	–	–L
Symbol	 Mi: Minuend word Su: Subtrahend word R: Result word	 Mi: Minuend word Su: Subtrahend word R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		–	–L	–	–L
Mi	–: Minuend word –L: First minuend word	INT	DINT	1	2
Su	–: Subtrahend word –L: First subtrahend word	INT	DINT	1	2
R	–: Result word –L: First result word	INT	DINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
–	MI, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
–L	Mi, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	OK	---	---	---	---
	R											---							

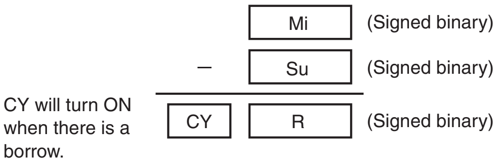
Flags

Name	Label	Operation	
		–	**–L**
Error Flag	ER	OFF	OFF
Equals Flag	=	• ON when the result is 0. • OFF in all other cases.	• 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.	• 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.	• 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 negative number from a positive number is in the range 0000 to 7FFF hex. • OFF in all other cases.	• 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.	• ON when the leftmost bit of the result is 1. • OFF in all other cases.

Function

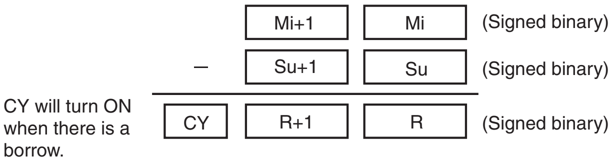
● –

–(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.



● –L

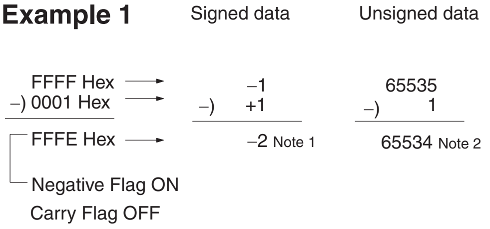
–L(411) subtracts the binary values in Su and Su+1 from Mi and Mi+1 and outputs the result to R, R+1. When the result is negative, it is output to R and R+1 as a 2's complement.



Hint

- **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.

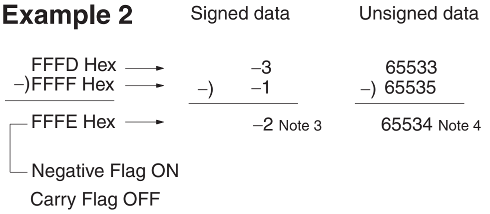
Example 1



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.

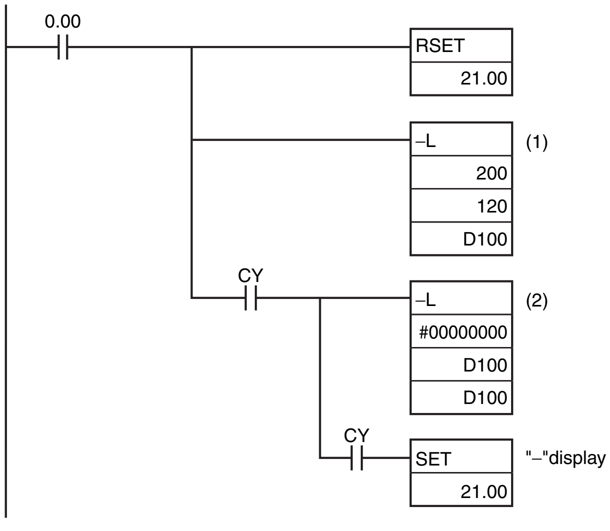
Example 2



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.

20F55A10 - B8A360E3 = -97AE06D3. (Hexadecimal)



In this example, the eight-digit binary value in CIO 121 and CIO 120 is subtracted from the value in CIO 201 and CIO 200, and the result is output in eight-digit binary to D101 and D100. If the result is negative, the instruction at (2) will be executed, and the actual result will then be output to D101 and D100.

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)

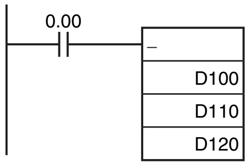
	0 0 0 0	0 0 0 0
	Su+1: D101	Su: D100
–	6 8 5 1	F 9 2 D
CY	R+1: D101	R+1: D100
1	9 7 A E	0 6 D 3

Final Subtraction Result

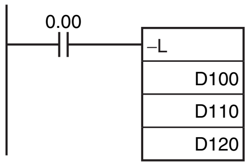
	Mi+1: CIO 201	Mi: CIO 200
	2 0 F 5	5 A 1 0
	Su+1: D101	Su: D100
–	6 8 5 1	F 9 2 D
CY	R+1: D101	R+1: D100
1	9 7 A E	0 6 D 3

The Carry Flag (CY) is turned ON, so the actual number is –97AE06D3. Because the content of D101 and D100 is negative, CY is used to turn ON CIO 21.00 to indicate this.

Example Programming



When CIO 0.00 is ON in the following example, D110 will be subtracted from D100 as 4-digit signed binary values and the result will be output to D120.

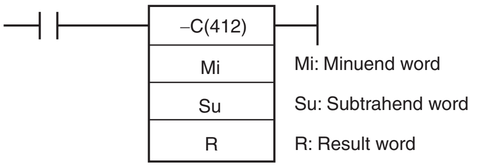
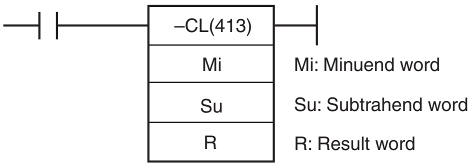


When CIO 0.00 is ON in the following example, D111 and D110 will be subtracted from D101 and D100 as 8-digit signed binary values and the result will be output to D121 and D120.

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.

–C/–CL

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY SUBTRACT WITH CARRY	–C	@–C	412	Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	@–CL	413	Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

	–C	–CL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		–C	–CL	–C	–CL
Mi	–C: Minuend word –CL: First minuend word	INT	DINT	1	2
Su	–C: Subtrahend word –CL: First subtrahend word	INT	DINT	1	2
R	–C: Result word –CL: First result word	INT	DINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
–C	MI, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
–CL	Mi, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

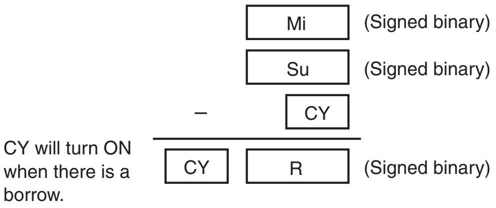
Flags

Name	Label	Operation	
		–C	–CL
Error Flag	ER	OFF	OFF
Equals Flag	=	• ON when the subtraction result is 0. • OFF in all other cases.	• 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.	• 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 8000 to FFFF hex. • OFF in all other cases.	• 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 0000 to 7FFF hex. • OFF in all other cases.	• 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.	• ON when the leftmost bit of the result is 1. • OFF in all other cases.

Function

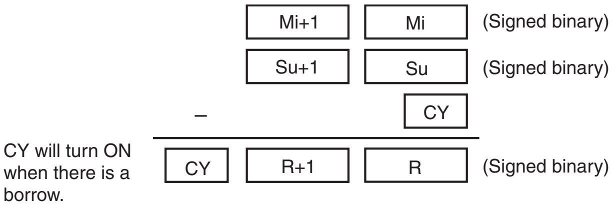
● –C

–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.



● –CL

–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.



Hint

- To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.
- **2's Complement**

  A 2's complement is the value obtained by subtracting each binary digit from 1 and adding one to the result.

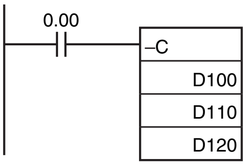
  Example: The 2's complement for the binary number 1101 is as follows:
  1111 (F hex) – 1101 (D hex) + 1 (1 hex) = 0011 (3 hex).

  Example: The 2's complement for the 4-digit hexadecimal number 3039 is as follows:
  FFFF hex - 3039 hex + 0001 hex = CFC7 hex.

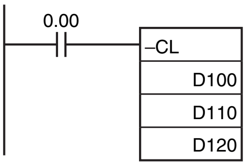
  Accordingly, the 2's complement for the 4-digit hexadecimal value "a" is as follows:
  FFFF hex – a hex + 0001 hex = b hex.
  And to obtain the true number "a" hex from the 2's complement "b" hex:
  a hex + 10000 hex – b hex.

  Example: To obtain the true number from the 2's complement CFC& hex:
  10000 hex – CFC7 hex = 3039 hex.

Example Programming



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.

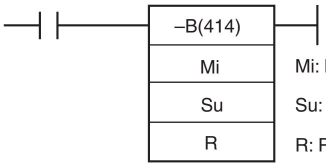
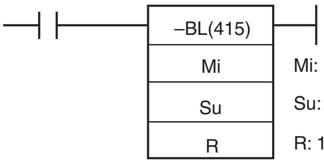


When CIO 0.00 is ON in the following example, D111, D110 and CY will be subtracted from D101 and D100 as 8-digit signed binary values, and the result will be output to D121 and D120.

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.

–B/–BL

Instruction	Mnemonic	Variations	Function code	Function
BCD SUBTRACT WITHOUT CARRY	–B	@–B	414	Subtracts 4-digit (single-word) BCD data and/or constants.
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	@–BL	415	Subtracts 8-digit (double-word) BCD data and/or constants.

	–B	–BL
Symbol	 Mi: Minuend word Su: Subtrahend word R: Result word	 Mi: 1st minuend word Su: 1st subtrahend word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		–B	–BL	–B	–BL
Mi	–B: Minuend word –BL: First minuend word	WORD	DWORD	1	2
Su	–B: Subtrahend word –BL: First subtrahend word	WORD	DWORD	1	2
R	–B: Result word –BL: First result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
–B	MI, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
–BL	Mi, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

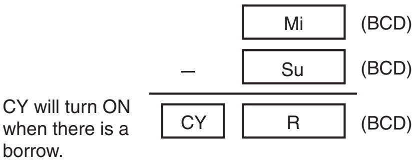
Flags

Name	Label	Operation	
		–B	–BL
Error Flag	ER	• ON when Mi is not BCD. • ON when Su is not BCD. • OFF in all other cases.	• 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.	• 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.	• ON when the subtraction results in a borrow. • OFF in all other cases.

Function

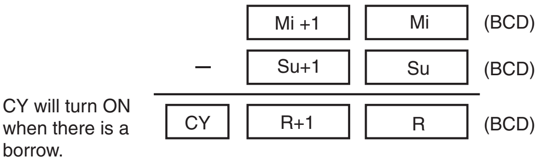
● –B

–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.



● –BL

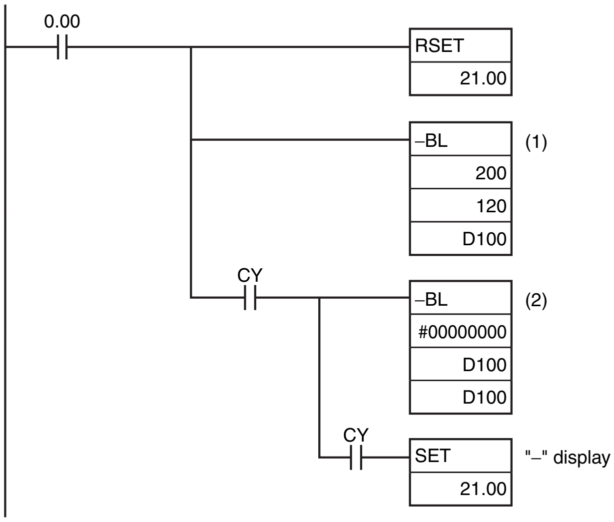
–BL(415) subtracts the BCD values in Su and Su+1 from Mi and Mi+1 and outputs the result to R, R+1. If the result is negative, it is output to R, R+1 as a 10's complement.



Hint

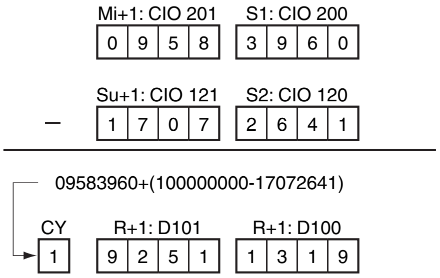
- **10's Complement**
  A 10's complement is the value obtained by subtracting each digit from 9 and adding one to the result. For example, the 10's complement for 7556 is calculated as follows: 9999 - 7556 + 1 = 2444. For a four digit number, the 10's complement of A is 9999 – A + 1 = B. To obtain the 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.

Example: 9,583,960 – 17,072,641 = -7,488,681. (BCD)



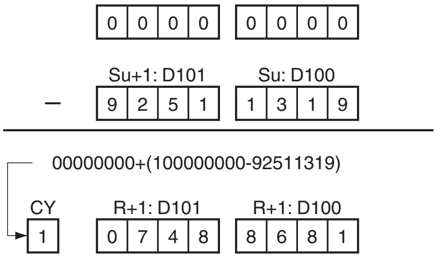
In this example, the eight-digit BCD content of CIO 121 and CIO 120 is subtracted from the content of CIO 201 and CIO 200, and the result is output in eight-digit BCD to D101 and D100. The result is negative, so the instruction at (2) will be executed, and the true value will then be output to D101 and D100.

Subtraction at (1)

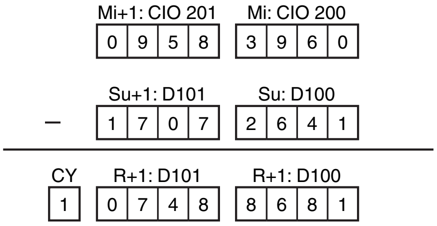


The Carry Flag (CY) is ON, so the result is subtracted from 0000 0000.

Subtraction at (2)

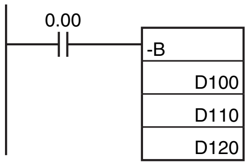


Final Subtraction Result

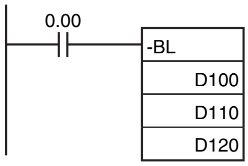


The Carry Flag (CY) will be turned ON, so the actual number is – 7,488,681. Because the content of D101 and D100 is negative, CY is used to turn ON CIO 21.00 to indicate this.

Example Programming



When CIO 0.00 is ON in the following example, D110 is subtracted from D100 as 4-digit BCD values, and the result will be output to D120.

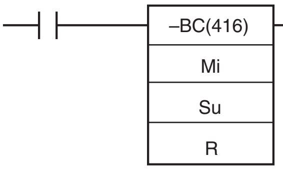
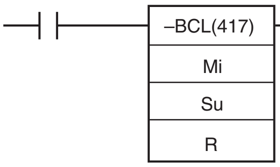


When CIO 0.00 is ON in the following example, D111 and D110 will be subtracted from D101 and D100 as 8-digit BCD values, and the result will be output to D121 and D120.

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.

–BC/–BCL

Instruction	Mnemonic	Variations	Function code	Function
BCD SUBTRACT WITH CARRY	–BC	@–BC	416	Subtracts 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	@–BCL	417	Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).

	–BC	–BCL
Symbol	 Mi: Minuend word Su: Subtrahend word R: Result word	 Mi: 1st minuend word Su: 1st subtrahend word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		–BC	–BCL	–BC	–BCL
Mi	–BC: Minuend word –BCL: First minuend word	WORD	DWORD	1	2
Su	–BC: Subtrahend word –BCL: First subtrahend word	WORD	DWORD	1	2
R	–BC: Result word –BCL: First result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
–BC	MI, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
–BCL	Mi, Su	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

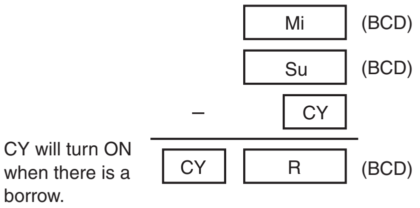
Flags

Name	Label	Operation	
		–BC	–BCL
Error Flag	ER	• ON when Mi is not BCD. • ON when Su is not BCD. • OFF in all other cases.	• 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.	• 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.	• ON when the subtraction results in a borrow. • OFF in all other cases.

Function

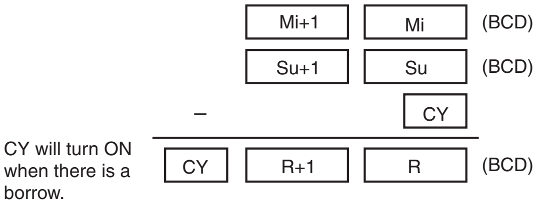
● –BC

–BC(416) subtracts BCD values in Su and CY from Mi and outputs the result to R. If the result is negative, it is output to R as a 10's complement.



● –BCL

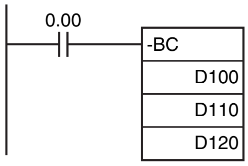
–BCL(417)subtracts the BCD values in Su, Su+1, and CY from Mi and Mi+1 and outputs the result to R, R+1. If the result is negative, it is output to R, R+1 as a 10's complement.



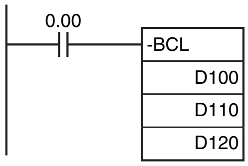
Hint

- To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.
- **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.

Example Programming



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.

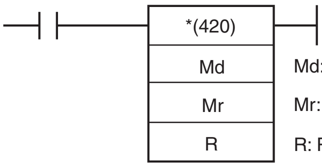
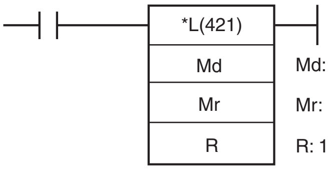


When CIO 0.00 is ON in the following example, D111, D110, and CY will be subtracted from D101 and D100 as 8-digit BCD values, and the result will be output to D121 and D120.

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.

*/*L

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY MULTIPLY	*	@*	420	Multiplies 4-digit signed hexadecimal data and/or constants.
DOUBLE SIGNED BINARY MULTIPLY	*L	@*L	421	Multiplies 8-digit signed hexadecimal data and/or constants.

	*	*L
Symbol	 Md: Multiplicand word Mr: Multiplier word R: Result word	 Md: 1st multiplicand word Mr: 1st multiplier word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		*	*L	*	*L
Md	*: Multiplicand word *L: First multiplicand word	INT	DINT	1	2
Mr	*: Multiplier word *L: First multiplier word	INT	DINT	1	2
R	First result word	DINT	LINT	2	4

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
*	Md, Mr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---	---						
*L	Md, Mr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

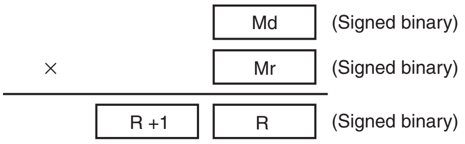
Flags

Name	Label	Operation	
		*	*L
Error Flag	ER	OFF	OFF
Equals Flag	=	• ON when the result is 0. • OFF in all other cases.	• 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.	• ON when the leftmost bit of the result is 1. • OFF in all other cases.

Function

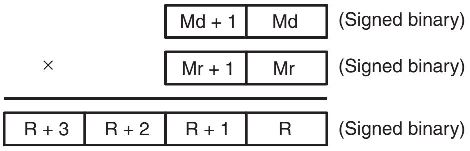
- *****

*(420) multiplies the signed binary values in Md and Mr and outputs the result to R, R+1.

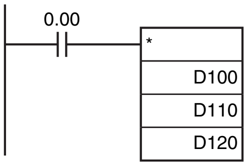


- ***L**

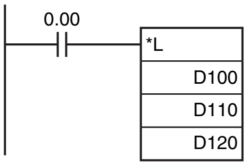
*L(421) multiplies the signed binary values in Md and Md+1 and Mr and Mr+1 and outputs the result to R, R+1, R+2, and R+3.



Example Programming



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 D120 and D121.

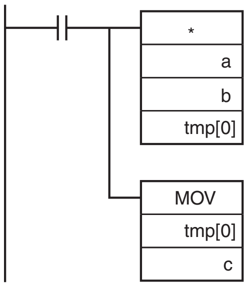


When CIO 0.00 is ON in the following example, D101, D100, D111, and D110 will be multiplied as 8-digit signed hexadecimal values and the result will be output to D123, D122, D121 and D120.

Example in Function Block Definition

In the following example, an array variable is used to get the result from the function block as one word.

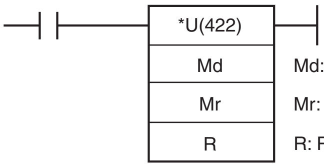
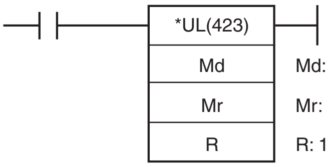
a * b → c



Function Block Variables
Multiplicand: a (data type: INT)
Multiplier: b (data type: INT)
Result: c (data type: INT)
Temporary variable: tmp (data type: WORD, 2-element array)

*U/*UL

Instruction	Mnemonic	Variations	Function code	Function
UNSIGNED BINARY MULTIPLY	*U	@*U	422	Multiplies 4-digit unsigned hexadecimal data and/or constants.
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	@*UL	423	Multiplies 8-digit unsigned hexadecimal data and/or constants.

	*U	*UL
Symbol	 Md: Multiplicand word Mr: Multiplier word R: Result word	 Md: 1st multiplicand word Mr: 1st multiplier word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		*U	*UL	*U	*UL
Md	*U: Multiplicand word *UL: First multiplicand word	UINT	UDINT	1	2
Mr	*U: Multiplier word *UL: First multiplier word	UINT	UDINT	1	2
R	First result word	UDINT	ULINT	2	4

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
*U	Md, Mr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---	---						
*UL	Md, Mr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

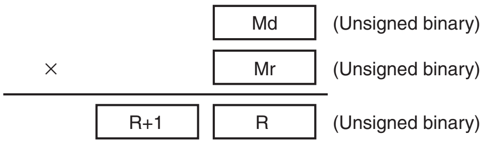
Flags

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.

Function

*U

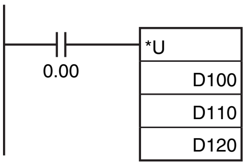
*U(420) multiplies the binary values in Md and Mr and outputs the result to R, R+1.



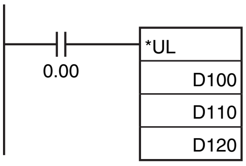
*UL

*UL(423) multiplies the unsigned 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.

Example Programming



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.

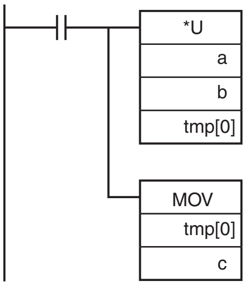


When CIO 0.00 is ON in the following example, D100, D110, D111, and D110 will be multiplied as 8-digit unsigned binary values and the result will be output to D123, D122, D121, and D120.

Example in Function Block Definition

In the following example, an array variable is used to get the result from the function block as one word.

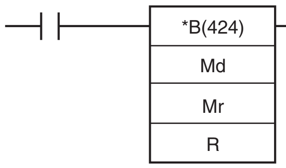
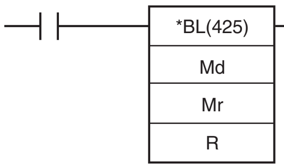
a * b → c



Function Block Variables
Multiplicand: a (data type: UINT)
Multiplier: b (data type: UINT)
Result: c (data type: UINT)
Temporary variable: tmp (data type: WORD, 2-element array)

*B/*BL

Instruction	Mnemonic	Variations	Function code	Function
BCD MULTIPLY	*B	@*B	424	Multiplies 4-digit (single-word) BCD data and/or constants.
DOUBLE BCD MULTIPLY	*BL	@*BL	425	Multiplies 8-digit (double-word) BCD data and/or constants.

	*B	*BL
Symbol	 Md: Multiplicand word Mr: Multiplier word R: Result word	 Md: 1st multiplicand word Mr: 1st multiplier word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		*B	*BL	*B	*BL
Md	*B: Multiplicand word *BL: First multiplicand word	WORD	DWORD	1	2
Mr	*B: Multiplier word *BL: First multiplier word	WORD	DWORD	1	2
R	First result word	DWORD	LWORD	2	4

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
*B	Md, Mr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---	---						
*BL	Md, Mr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

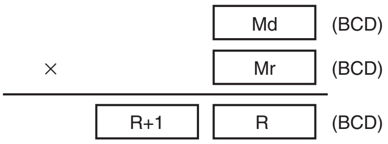
Flags

Name	Label	Operation	
		*B	*BL
Error Flag	ER	• ON when Md is not BCD. • ON when Mr is not BCD. • OFF in all other cases.	• 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.	• ON when the result is 0. • OFF in all other cases.

Function

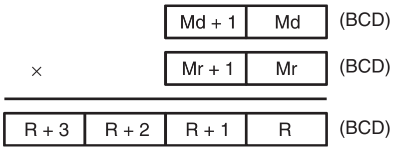
● *B

*B(424) multiplies the BCD content of Md and Mr and outputs the result to R, R+1.

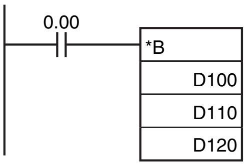


● *BL

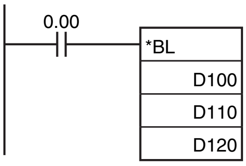
*BL(425) multiplies BCD values in Md and Md+1 and Mr and Mr+1 and outputs the result to R, R+1, R+2, and R+3.



Example Programming



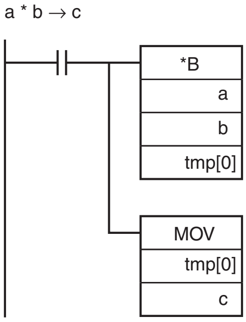
When CIO 0.00 is ON in the following example, D100 and D110 will be multiplied as 4-digit BCD values and the result will be output to D121 and D120.



When CIO 0.00 is ON in the following example, D101, D100, D111, and D110 will be multiplied as 8-digit unsigned BCD values and the result will be output to D123, D122, D121 and D120.

● Example in Function Block Definition

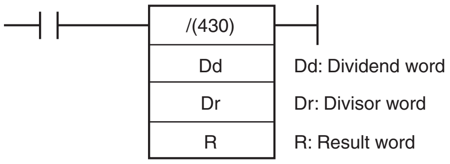
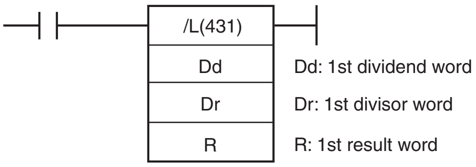
In the following example, an array variable is used to get the result from the function block as one word.



Function Block Variables
  Multiplicand: a (data type: WORD)
  Multiplier: b (data type: WORD)
  Result: c (data type: WORD)
  Temporary variable: tmp (data type: WORD, 2-element array)

/, /L

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY DIVIDE	/	@/	430	Divides 4-digit (single-word) signed hexadecimal data and/or constants.
DOUBLE SIGNED BINARY DIVIDE	/L	@/L	431	Divides 8-digit (double-word) signed hexadecimal data and/or constants.

	/	/L
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		/	/L	/	/L
Dd	/: Dividend word /L: First dividend word	INT	DINT	1	2
Dr	/: Divisor word /L: First divisor word	INT	DINT	1	2
R	First result word	DWORD	LWORD	2	4

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
/	Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---	---						
/L	Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

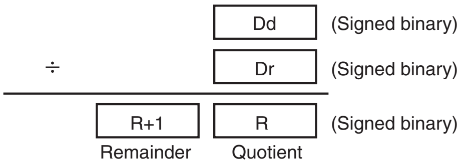
Flags

Name	Label	Operation
Error Flag	ER	• ON when the divisor is 0. • OFF in all other cases.
Equals Flag	=	• ON when as a result of the division, R/R+1, R is 0. • OFF in all other cases.
Negative Flag	N	• ON when the leftmost bit of the R/R+1, R is 1. • OFF in all other cases.

Function

● /

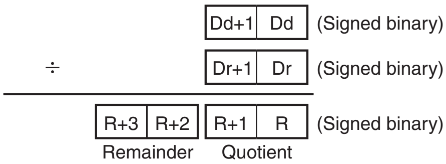
/(430) divides the signed binary (16 bit) values in Dd by those in Dr and outputs the result to R, R+1. The quotient is placed in R and the remainder in R+1.



Note Division of hexadecimal #8000 by #FFFF is undefined.

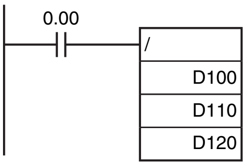
● /L

/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.

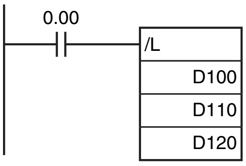


Note Division of hexadecimal #80000000 by #FFFFFFFF is undefined.

Example Programming



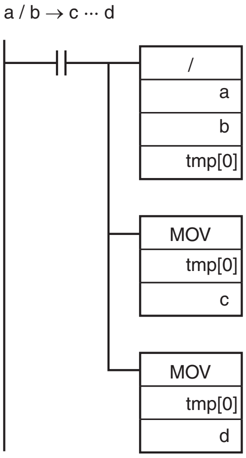
When CIO 0.00 is ON in the following example, D100 will be divided by D110 as 4-digit signed binary values and the quotient will be output to D120 and the remainder to D121.



When CIO 0.00 is ON in the following example, D101 and D100 are divided by D111 and D110 as 8-digit signed hexadecimal values and the quotient will be output to D121 and D120 and the remainder to D123 and D122.

● Example in Function Block Definition

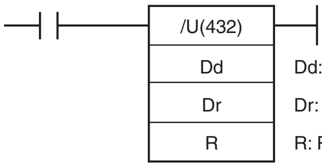
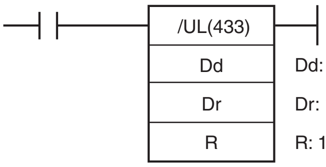
In the following example, an array variable is used to get the quotient and remainder from the function block.



Function Block Variables
- Dividend: a (data type: INT)
- Divisor: b (data type: INT)
- Quotient: c (data type: INT)
- Remainder: d (data type: INT)
- Temporary variable: tmp (data type: WORD, 2-element array)

/U, /UL

Instruction	Mnemonic	Variations	Function code	Function
UNSIGNED BINARY DIVIDE	/U	@/U	432	Divides 4-digit (single-word) unsigned hexadecimal data and/or constants.
DOUBLE UNSIGNED BINARY DIVIDE	/UL	@/UL	433	Divides 8-digit (double-word) unsigned hexadecimal data and/or constants.

	/U	/UL
Symbol	 Dd: Dividend word Dr: Divisor word R: Result word	 Dd: 1st dividend word Dr: 1st divisor word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		/U	/UL	/U	/UL
Dd	/U: Dividend word /UL: First dividend word	UINT	UDINT	1	2
Dr	/U: Divisor word /UL: First divisor word	UINT	UDINT	1	2
R	First result word	DWORD	LWORD	2	4

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
/U	Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---	---						
/UL	Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

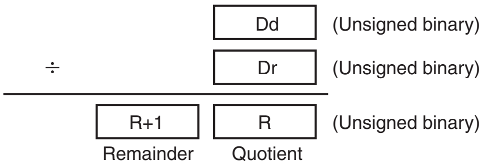
Flags

Name	Label	Operation
Error Flag	ER	• ON when the divisor is 0. • OFF in all other cases.
Equals Flag	=	• ON when as a result of the division R/R+1, R is 0. • OFF in all other cases.
Negative Flag	N	• ON when the leftmost bit of the R/R+1, R is 1. • OFF in all other cases.

Function

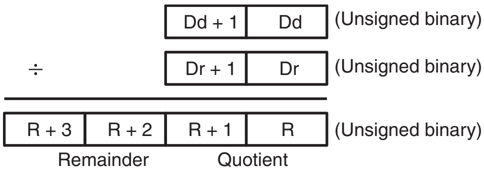
● /U

/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.

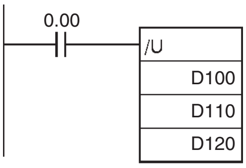


● /UL

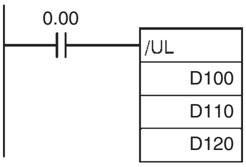
/UL(433) divides the unsigned binary values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the quotient to R, R+1 and the remainder to R+2, and R+3.



Example Programming



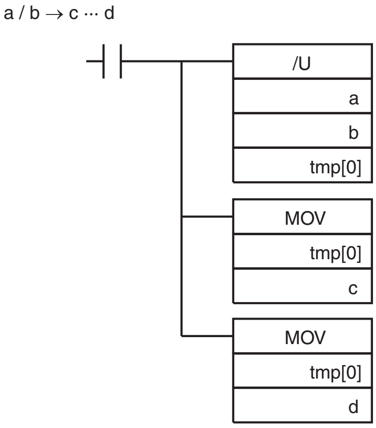
When CIO 0.00 is ON in the following example, D100 will be divided by D110 as 4-digit unsigned binary values and the quotient will be output to D120 and the remainder will be output to D121.



When CIO 0.00 is ON in the following example, D100 and D101 will be divided by D111 and D110 as 8-digit unsigned hexadecimal values and the quotient will be output to D121 and D120 and the remainder to D123 and D122.

● Example in Function Block Definition

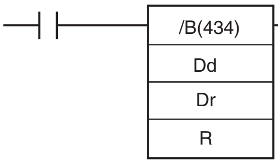
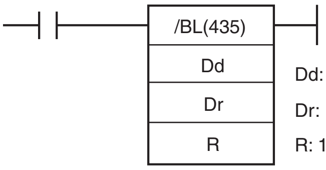
In the following example, an array variable is used to get the quotient and remainder from the function block.



Function Block Variables
Dividend: a (data type: UINT)
Divisor: b (data type: UINT)
Quotient: c (data type: UINT)
Remainder: d (data type: UINT)
Temporary variable: tmp (data type: WORD, 2-element array)

/B, /BL

Instruction	Mnemonic	Variations	Function code	Function
BCD DIVIDE	/B	@/B	434	Divides 4-digit (single-word) BCD data and/or constants.
DOUBLE BCD DIVIDE	/BL	@/BL	435	Divides 8-digit (double-word) BCD data and/or constants.

	/B	/BL
Symbol	 Dd: Dividend word Dr: Divisor word R: Result word	 Dd: 1st dividend word Dr: 1st divisor word R: 1st result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		/B	/BL	/B	/BL
Dd	/B: Dividend word /BL: First dividend word	WORD	DWORD	1	2
Dr	/B: Divisor word /BL: First divisor word	WORD	DWORD	1	2
R	First result word	DWORD	LWORD	2	4

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
*B	Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---	---						
*BL	Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

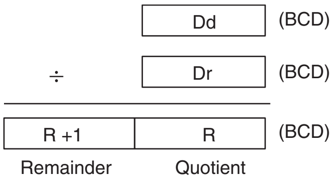
Flags

Name	Label	Operation	
		/B	/BL
Error Flag	ER	• ON when Dd is not BCD. • ON when Dr is not BCD. • ON when the divisor is 0. • OFF in all other cases.	• ON when Dd, Dd+1 is not BCD. • ON when Dr, Dr +1 is not BCD. • ON when the divisor 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.	• ON when as a result of the division R+1, R is 0. • OFF in all other cases.

Function

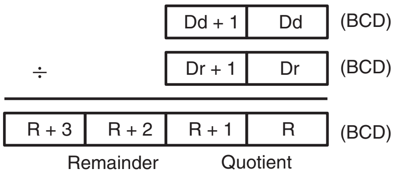
● /B

/B(434) divides the BCD content of Dd by those of Dr and outputs the quotient to R and the remainder to R+1.

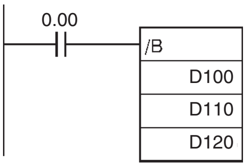


● /BL

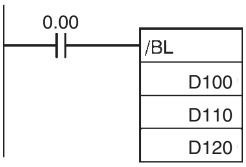
/BL(435) divides BCD values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the quotient to R, R+1 and the remainder to R+2, R+3.



Example Programming



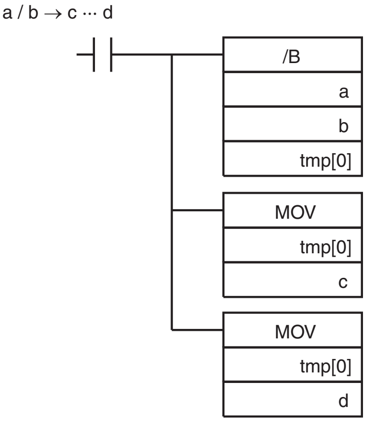
When CIO 0.00 is ON in the following example, D100 will be divided by D110 as 4-digit BCD values and the quotient will be output to D120 and the remainder to D121.



When CIO 0.00 is ON in the following example, D101 and D100 will be divided by D111 and D110 as 8-digit BCD values and the quotient will be output to D121 and D120 and the remainder to D123 and D122.

● Example in Function Block Definition

In the following example, an array variable is used to get the quotient and remainder from the function block.

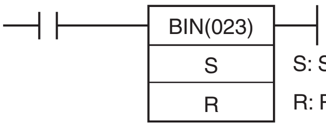
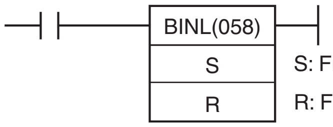


Function Block Variables
Dividend: a (data type: WORD)
Divisor: b (data type: WORD)
Quotient: c (data type: WORD)
Remainder: d (data type: WORD)
Temporary variable: tmp (data type: WORD, 2-element array)

Conversion Instructions

BIN/BINL

Instruction	Mnemonic	Variations	Function code	Function
BCD TO BINARY	BIN	@BIN	023	Converts BCD data to binary data.
DOUBLE BCD TO DOUBLE BINARY	BINL	@BINL	058	Converts 8-digit BCD data to 8-digit hexadecimal (32-bit binary) data.

	BIN	BINL
Symbol	 S: Source word R: Result word	 S: First source word R: First result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		BIN	BINL	BIN	BINL
S	BIN: Source word BINL: First source word	WORD	DWORD	1	2
R	BIN: Results word BINL: First result word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
BIN	S R	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
BINL	S R	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

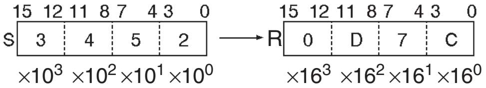
Name	Label	Operation	
		BIN	BINL
Error Flag	ER	• ON if the content of S is not BCD. • OFF in all other cases.	• ON if the contents of S+1, S are not BCD. • OFF in all other cases.
Equals Flag	=	• ON if the result is 0000. • OFF in all other cases.	• ON if the result is 0. • OFF in all other cases.
Negative Flag	N	OFF	OFF

Function

● BIN

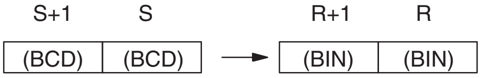
BIN(023) converts the BCD data in S to binary data and writes the result to R.

The following diagram shows an example BCD-to-binary conversion.

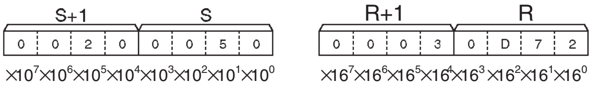


● BINL

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.

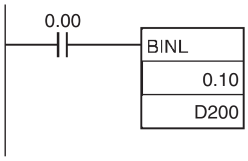


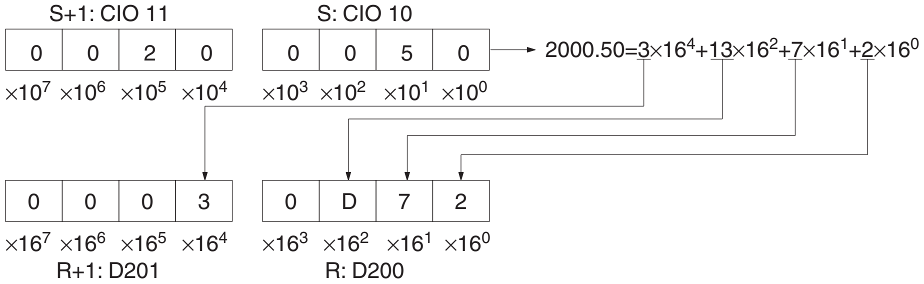
The following diagram shows an example of 8-digit BCD-to-binary conversion.



Example Programming

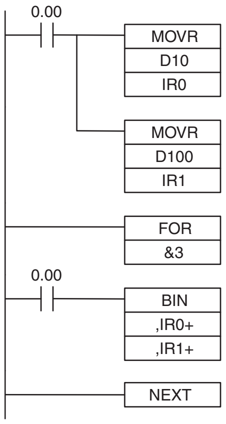
When CIO 0.00 is ON in the following example, the 8-digit BCD value in CIO 0010 and CIO 0011 is converted to hexadecimal and stored in D200 and D201.

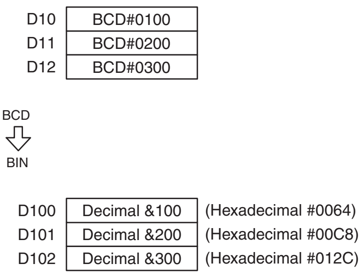




● In this example, N words of BCD data is converted to binary data.

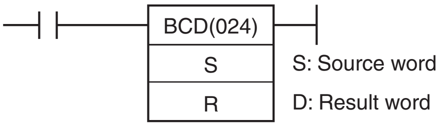
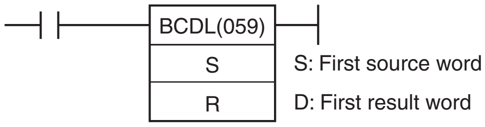
If N = 3, the three words of BCD starting from D10 will be converted to binary data one word at a time when CIO 0.00 turns ON. The resulting binary data will be stored starting from D100.





BCD/BCDL

Instruction	Mnemonic	Variations	Function code	Function
BINARY TO BCD	BCD	@BCD	024	Converts a word of binary data to a word of BCD data.
DOUBLE BINARY TO DOUBLE BCD	BCDL	@BCDL	059	Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data.

	BCD	BCDL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		BCD	BCDL	BCD	BCDL
S	BCD: Source word BCDL: First source word	UINT	UDINT	1	2
R	BCD: Result word BCDL: First result word	WORD	DWORD	1	2

S: Source Word (BCD)/First Source Word (BCDL)

- BCD
  S must be between 0000 and 270F hexadecimal (0000 and 9999 decimal).
- BCDL
  The content of S+1 and S must be between 0000 0000 and 05F5 E0FF hexadecimal (0000 0000 and 9999 9999 decimal).

Note S to S+1 and D to D+1 must each be the same area type.

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
BCD	S R	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
BCDL	S R	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

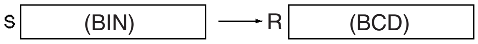
Flags

Name	Label	Operation	
		BCD	BCDL
Error Flag	ER	• ON if the content of S exceeds 270F (9999 decimal). • OFF in all other cases.	• 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 0000. • OFF in all other cases.	• ON if the result is 0. • OFF in all other cases.

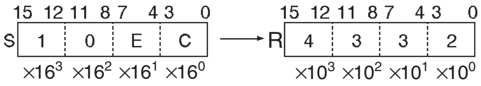
Function

● BCD

BCD(024) converts the binary data in S to BCD data and writes the result to R.

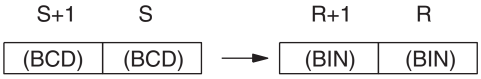


The following diagram shows an example BCD-to-binary conversion.

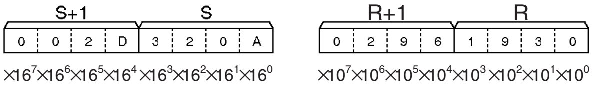


● BCDL

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.

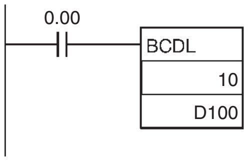


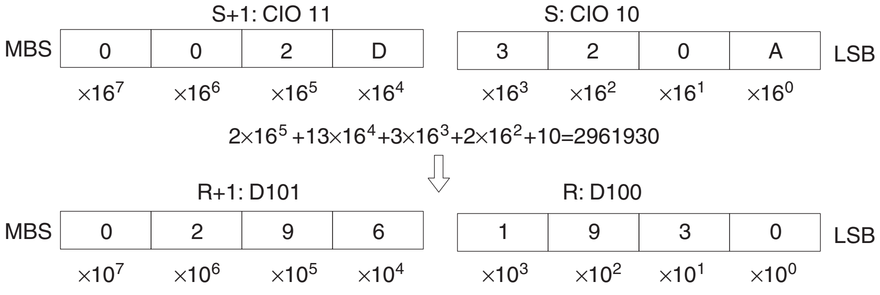
The following diagram shows an example of 8-digit BCD-to-binary conversion.



Example Programming

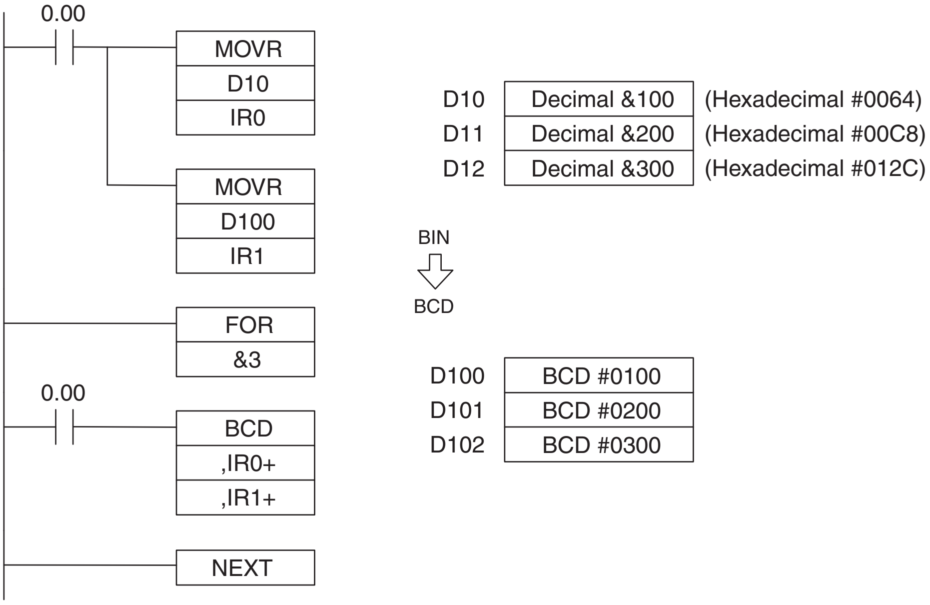
When CIO 0.00 is ON in the following example, the hexadecimal value in CIO 11 and CIO 10 is converted to a BCD value and stored in D100 and D101.





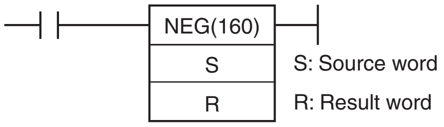
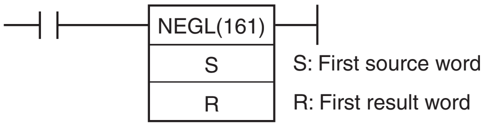
● In this example, N words of binary data is converted to BCD data.

If N = 3, the three words of binary starting from D10 will be converted to binary data one word at a time when CIO 0.00 turns ON. The resulting BCD data will be stored starting from D100.



NEG/NEGL

Instruction	Mnemonic	Variations	Function code	Function
2'S COMPLEMENT	NEG	@NEG	160	Calculates the 2's complement of a word of hexadecimal data.
DOUBLE 2'S COMPLEMENT	NEGL	@NEGL	161	Calculates the 2's complement of two words of hexadecimal data.

	NEG	NEGL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		NEG	NEGL	NEG	NEGL
S	NEG: Source word NEGL: First source word	WORD	DWORD	1	2
R	NEG: Result word NEGL: First result word	UINT	UDINT	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
NEG	S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
NEGL	S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	• ON if the result is 0000/0000 0000. • OFF in all other cases.
Negative Flag	N	• ON if bit 15 of the result is ON. • OFF in all other cases.

Function

● NEG

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.

$$\overline{(S)} \xrightarrow{\text{2's complement (Complement + 1)}} (R)$$

Note The result for 8000 hex will be 8000 hex.

● NEGL

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.

$$\overline{(S+1, S)} \xrightarrow{\text{2's complement (Complement + 1)}} (R+1, R)$$

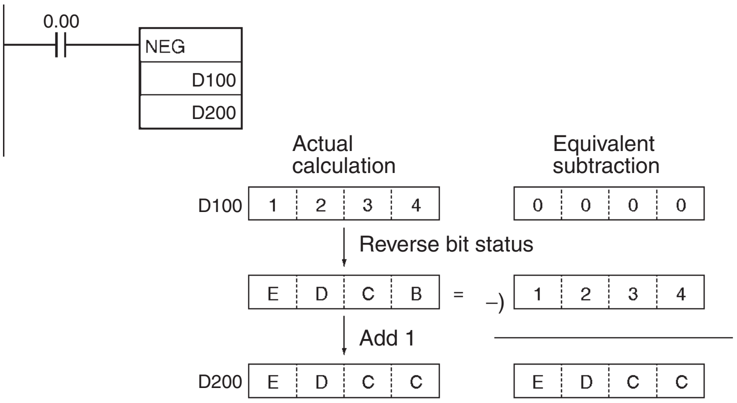
Note The result for 8000 0000 hex will be 8000 0000 hex.

Hint

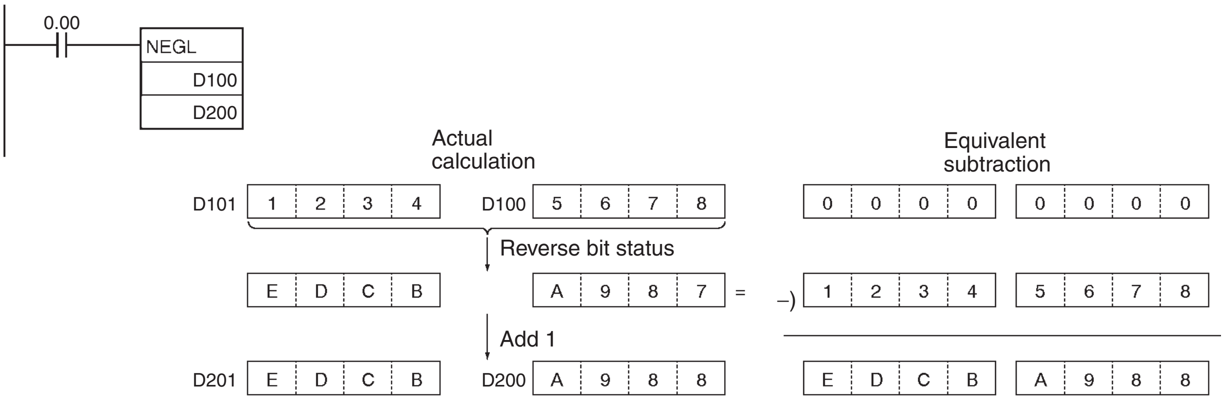
- This operation (reversing the status of the bits and adding 1) is equivalent to subtracting the content of S/S+1 and S from 0000/0000 0000.

Example Programming

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.

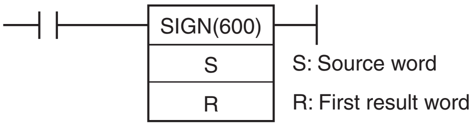


When CIO 0.00 is ON in the following example, NEGL(161) calculates the 2's complement of the content of D101 and D100 and writes the result to D201 and D200.



SIGN

Instruction	Mnemonic	Variations	Function code	Function
16-BIT TO 32-BIT SIGNED BINARY	SIGN	@SIGN	600	Expands a 16-bit signed binary value to its 32-bit equivalent.

	SIGN
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
R	First result word	DINT	2

R: First result word

R: Contents of S
R+1: #0000 or #FFFF (hex)

Note R and R+1 must be in the same data area.

● Operand Specifications

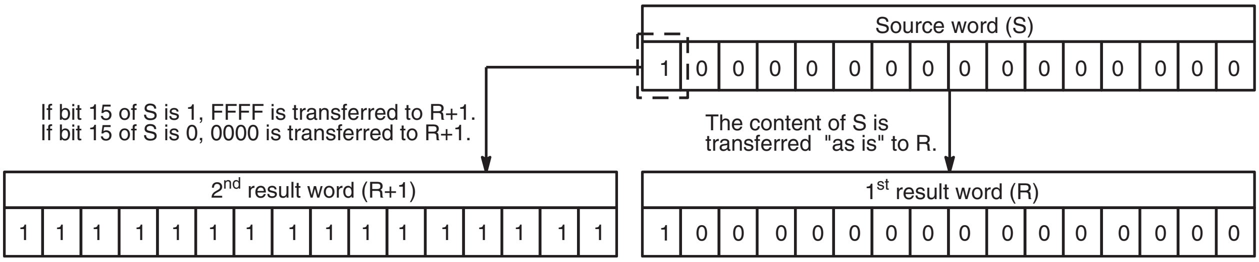
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
R											---	---						

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.

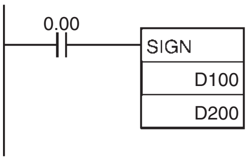
Function

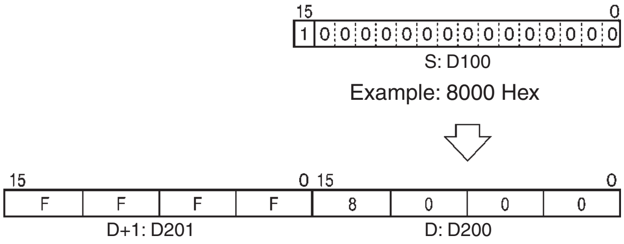
SIGN(600) converts the 16-bit signed binary number in S to its 32-bit signed binary equivalent and writes the result in R+1 and R.
The conversion is accomplished by copying the content of S to R and writing FFFF to R+1 if bit 15 of S is 1 or writing 0000 to R+1 if bit 15 of S is 0.



Example Programming

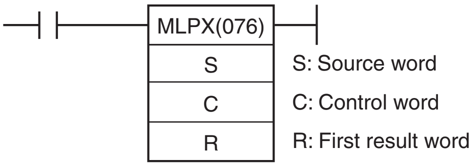
When CIO 0.00 is ON in the following example, SIGN(600) converts the 16-bit signed binary content of D100 (#8000 = -32,768 decimal) to its 32-bit equivalent (#FFFF 8000 = -32,768 decimal) and writes that result to D201 and D200.





MLPX

Instruction	Mnemonic	Variations	Function code	Function
DATA DECODER	MLPX	@MLPX	076	Reads the numerical value in the specified digit (or byte) in the source word with 4-to 16 conversion (or 8-to-256 conversion), turns ON the corresponding bit in the result word, and turns OFF all other bits in the result word.

	MLPX
Symbol	

Applicable Program Areas

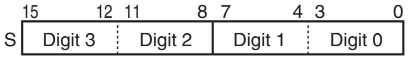
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	UINT	1
C	Control word	UINT	1
R	First result word	UINT	Variable

- **4-to-16 bit decoder**

S: Source Word



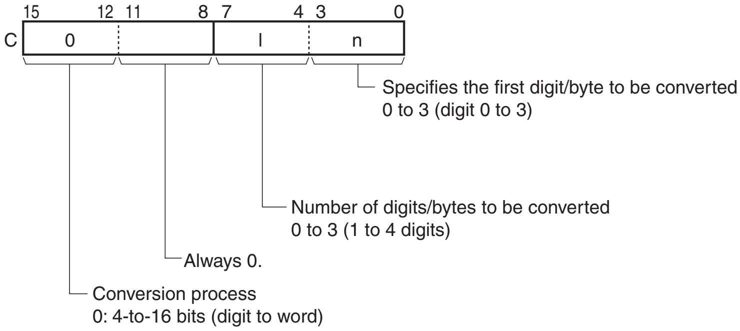
Digits from the starting digit going left are decoded
(Returns to digit 3 after digit 0)

R: First Result Word

D: Decoding result of 1st digit of decoded digits
D+1: Decoding result of 2nd digit of decoded digits
D+2: Decoding result of 3rd digit of decoded digits
D+3: Decoding result of 4th digit of decoded digits

Note The result words must be in the same data area.

C: Control Word



Specifies the first digit/byte to be converted
0 to 3 (digit 0 to 3)

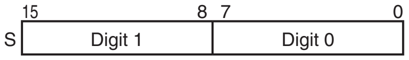
Number of digits/bytes to be converted
0 to 3 (1 to 4 digits)

Always 0.

Conversion process
0: 4-to-16 bits (digit to word)

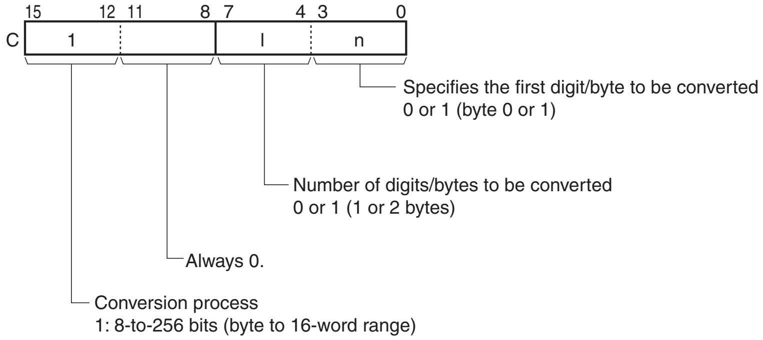
8-to-256 bit conversion

S: Source Word



Digits from the starting digit going left are decoded
(Returns to digit 0 after digit 1)

C: Control Word



R: First Result Word

D+15 to D: Decoding result of 1st digit of decoded digits
D+31 to D+16:
Decoding result of 2nd digit of decoded digits

Note The result words must be in the same data area.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
C											OK							
R											---	---						

Flags

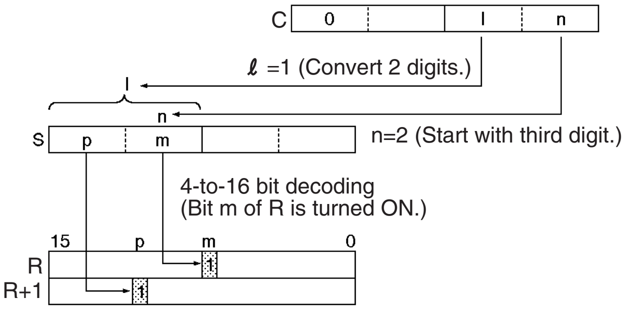
Name	Label	Operation
Error Flag	ER	• ON if C is not within the specified ranges. • OFF in all other cases.

Function

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.

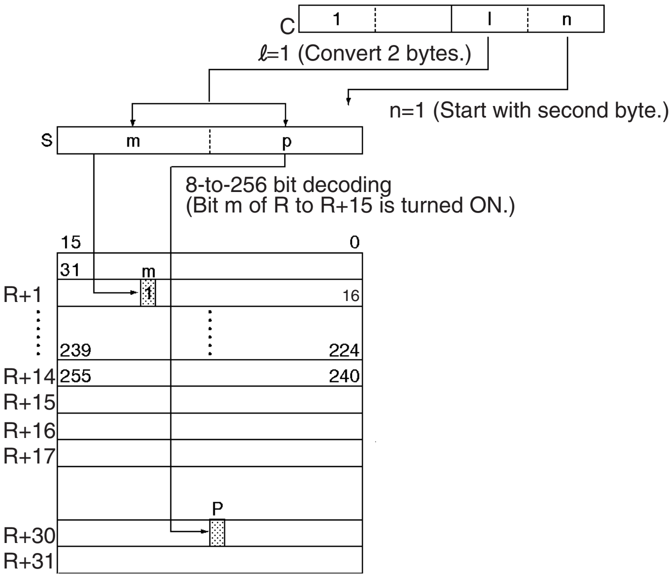
4-to-16 bit Conversion

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.



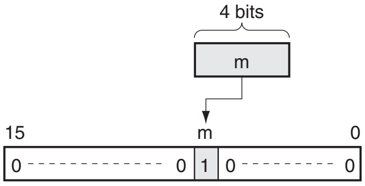
● 8-to-256 bit Conversion

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.

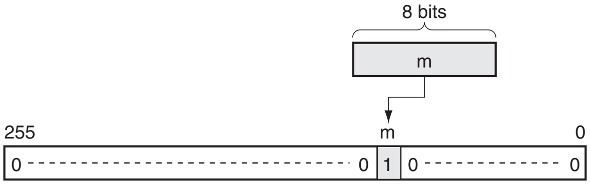


Hint

As shown at right, 4 to 16 decoding consists of taking the 4-bit binary value as the bit number and setting 1 in that bit number and 0 in the other bit numbers of the 16 bits.



As shown at right, 8 to 256 decoding consists of taking the 8-bit binary value as the bit number and setting 1 in that bit number and 0 in the other bit numbers of the 256 bits.



Precaution

● 4-to-16 bit conversion

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.

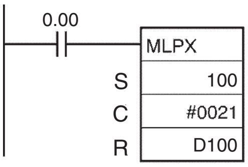
● 8-to-256 bit conversion

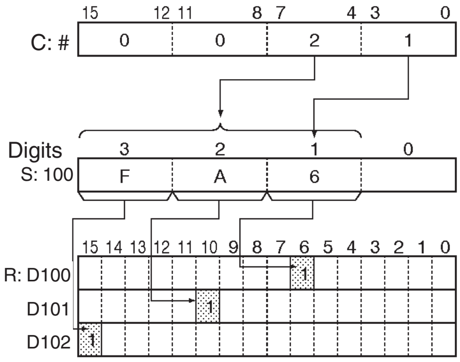
When two bytes are being converted, MLPX(076) will read the bytes in S from right to left and will wrap around to the rightmost byte if the leftmost byte (byte 1) has been specified as the starting byte.

Example Programming

● 4-to-16 bit Conversion

When CIO 0.00 is ON in the following example, MLPX(076) will convert 3 digits in S beginning with digit 1 (the second digit), as indicated by C (#0021). The corresponding bits in D100, D101, and D102 will be turned ON.





Bits 0 to 3: Starting digit (Digit 1)

Bits 4 to 7: Number of digits (3 digits)

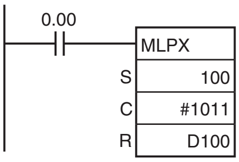
Digit 1 contains 6, so bit 6 is turned ON.

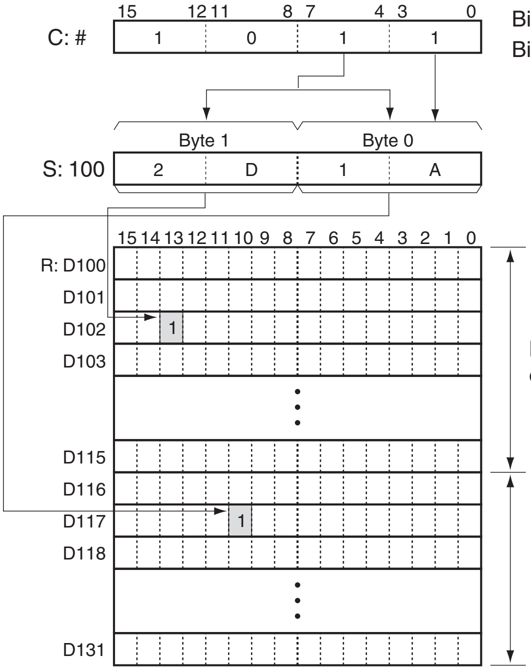
Digit 2 contains A, so bit 10 is turned ON.

Digit 3 contains F, so bit 15 is turned ON.

● 8-to-256 bit Conversion

When CIO 0.00 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 D100 to D115 and D116 to D131 will be turned ON.





Bits 0 to 3: Starting byte (Byte 1)

Bits 4 to 7: Number of bytes (2 bytes)

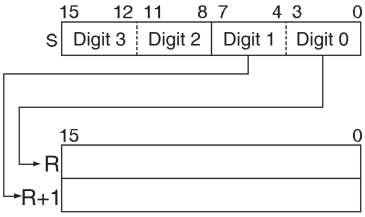
Byte 1 contains 2D, so bit 13 (D) of R+2 is turned ON.

Byte 0 contains 1A, so bit 10 (A) of R+1 is turned ON.

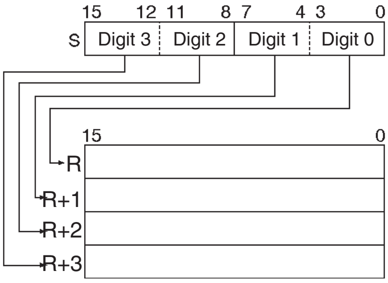
● Example of multi-digit decoding

- Example of 4-to-16 bit decoding

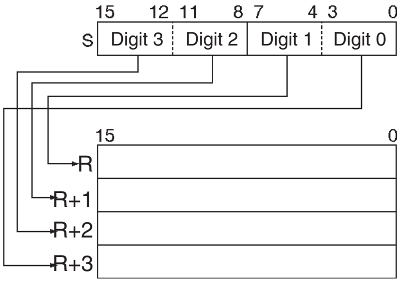
C: #0010



C: #0030

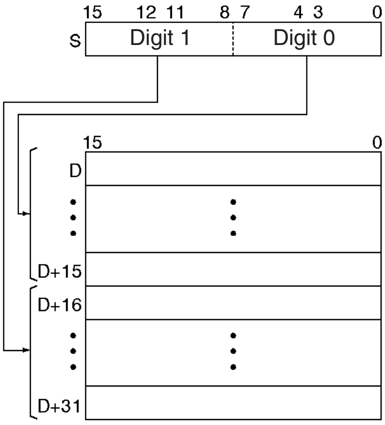


C: #0031

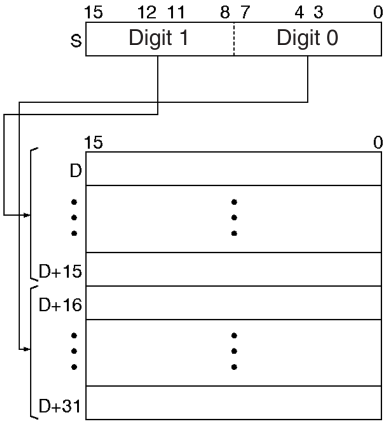


- Example of 8-to-256 bit decoding

C: #1010

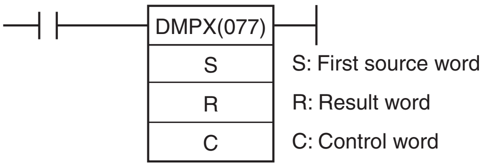


C: #1011



DMPX

Instruction	Mnemonic	Variations	Function code	Function
DATA ENCODER	DMPX	@DMPX	077	Finds the location of the first or last ON bit within the source word with 16-to-4 conversion (or 256-to-8 conversion), and writes that value to the specified digit (or byte) in the result word.

	DMPX
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

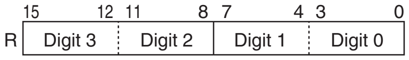
Operand	Description	Data type	Size
S	First source word	UINT	Variable
R	Result word	UINT	1
C	Control word	UINT	1

- **16-to-4 bit conversion**

S: First Source Word

S: 1st digit of digits to be encoded
S+1: 2nd digit of digits to be encoded
S+2: 3rd digit of digits to be encoded
S+3: 4th digit of digits to be encoded

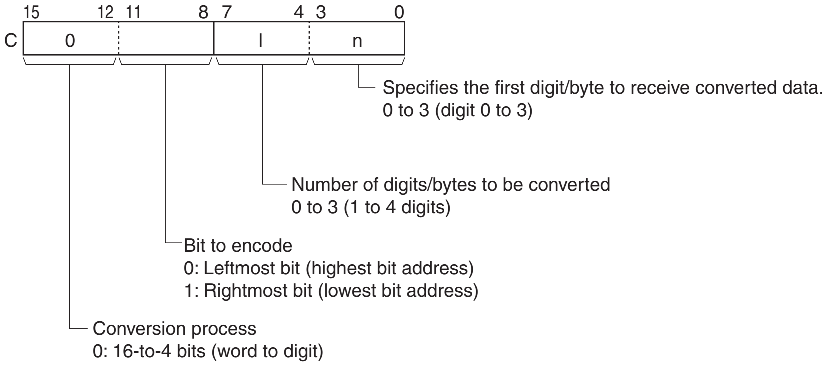
R: Result Word



The results of encoding of S to S+3 are stored from the starting digit going left (returns to digit 0 after digit 3).

Note The source words must be in the same data area.

C: Control Word



Specifies the first digit/byte to receive converted data.
0 to 3 (digit 0 to 3)

Number of digits/bytes to be converted
0 to 3 (1 to 4 digits)

Bit to encode
0: Leftmost bit (highest bit address)
1: Rightmost bit (lowest bit address)

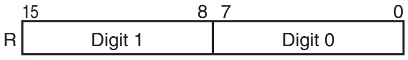
Conversion process
0: 16-to-4 bits (word to digit)

- **256-to-8 bit conversion**

S: First Source Word

S+15 to S: 1st digit of digits to be encoded
S+31 to S+16: 2nd digit of digits to be encoded

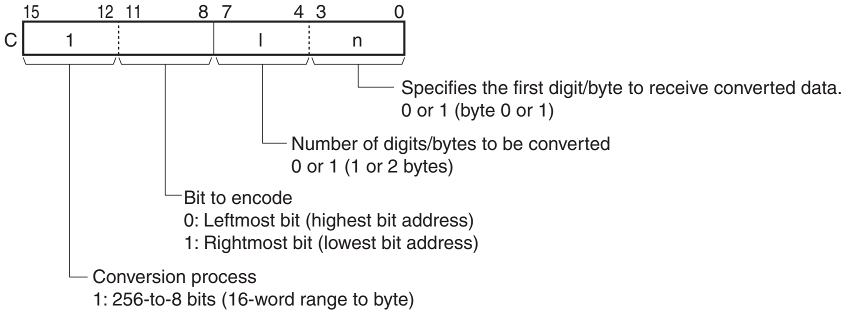
R: Result Word



The results of encoding of S to S+15, S+16 to S+31 are stored from the starting digit going left (returns to digit 0 after digit 1).

Note The source words must be in the same data area.

C: Control word



- **Operand Specifications**

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
R											---	OK						
C											OK	OK						

Flags

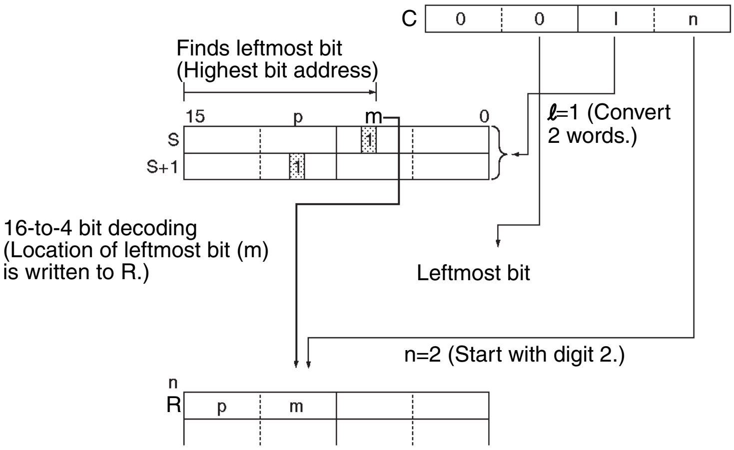
Name	Label	Operation
Error Flag	ER	• ON if any of the source words contains 0000 hex (i.e., no bit to encode). • ON if C is not within the specified ranges. • OFF in all other cases.

Function

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.

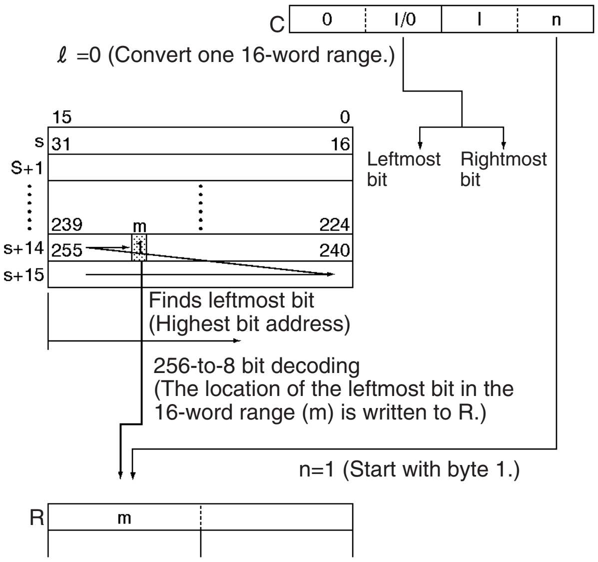
- **16-to-4 bit Conversion**

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.



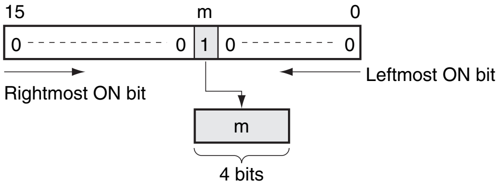
● 256-to-8 bit Conversion

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.

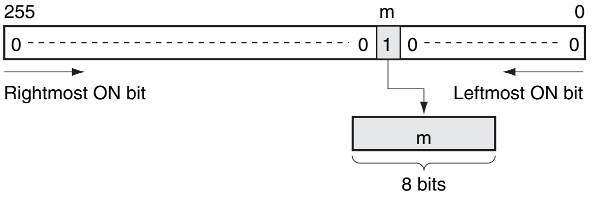


Hint

As shown at right, 16 to 4 encoding consists of converting the bit number (m) of the leftmost or rightmost bit that has 1 set among the 16 bits to a 4-bit binary value.



As shown at right, 256 to 8 encoding consists of converting the bit number (m) of the leftmost or rightmost bit that has 1 set among the 256 bits to an 8-bit binary value.



Precaution

● 16-to-4 bit conversion

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.

● 256-to-8 bit conversion

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.

● If one word is 0 (no bit to encode) in any of the conversion data specified in S (First Source Word), the error flag turns ON.

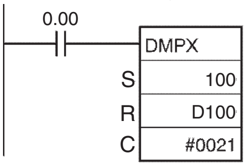
On the CS1G/H-CPU□□(-V1), the error flag turns ON, but Words that are not 0 in the conversion data specified in S are converted and output to the corresponding digits of R (Result Word).

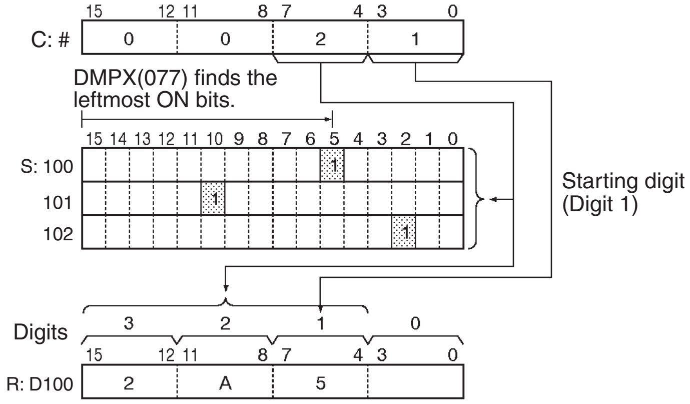
In models other than the above, the error flag turns ON and nothing is output to R.

Example Programming

● 16-to-4 bit Conversion

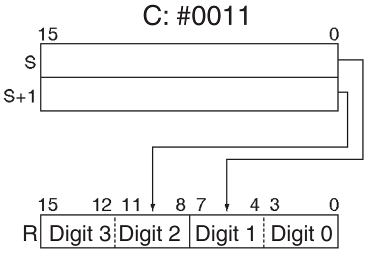
When CIO 0.00 is ON in the following example, DMPX(077) will find the leftmost ON bits in CIO 100, CIO 101, and CIO 102 and write those locations to 3 digits in R beginning with digit 1 (the second digit), as indicated by C (#0021).

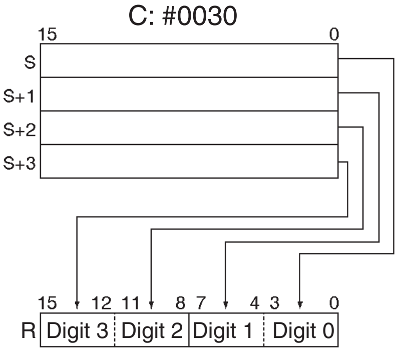


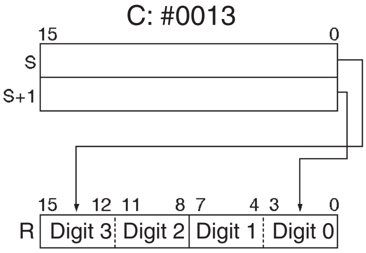


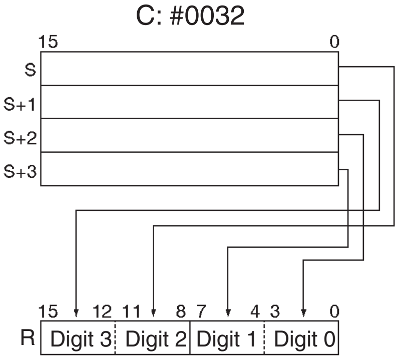
● Example of multi-digit decoding

- Example of 16-to-4 bit decoding



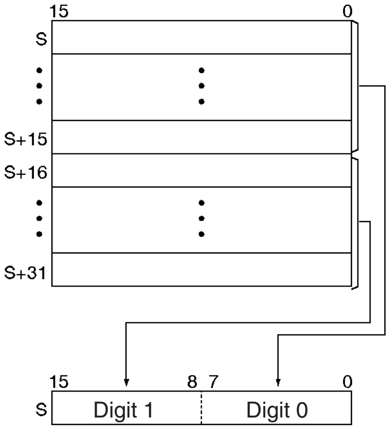




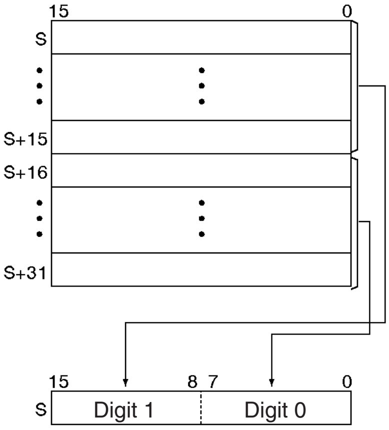


256-to-8 bit Conversion

C: #1010



C: #1011



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

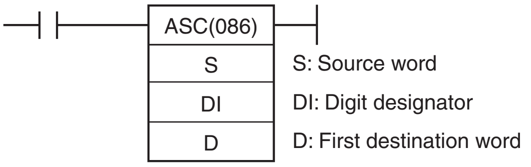
DMPX(077) D1 D100 #0001

DMPX(077) D2 D100 #0002

DMPX(077) D3 D100 #0003

ASC

Instruction	Mnemonic	Variations	Function code	Function
ASCII CONVERT	ASC	@ASC	086	Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents.

	ASC
Symbol	

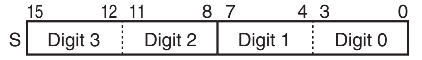
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

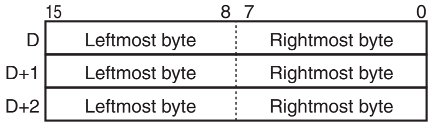
Operand	Description	Data type	Size
S	Source word	UINT	1
DI	Digit designator	UINT	1
D	First destination word	UINT	Variable

S: Source Word



Digits from the starting digit of conversion going left are treated as HEX data and converted to ASCII code (returns to digit 0 after digit 3)

D: First Destination Word

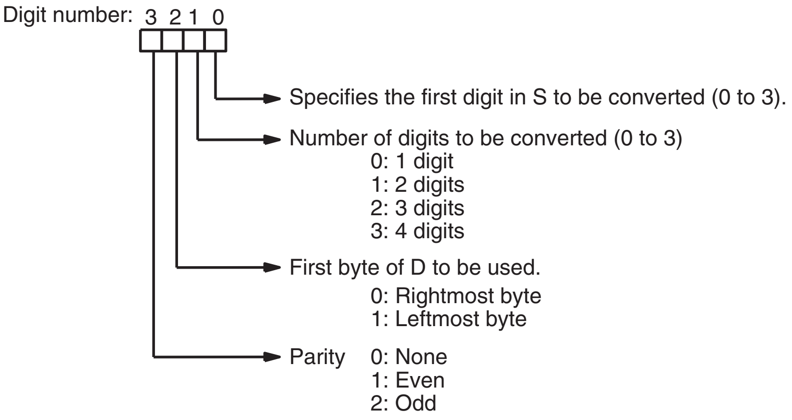


ASCII code is stored in the left word side. The code is stored from the output starting byte of D in the order rightmost byte, leftmost byte.

Note The destination 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.



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
DI											OK							
D											---	---						

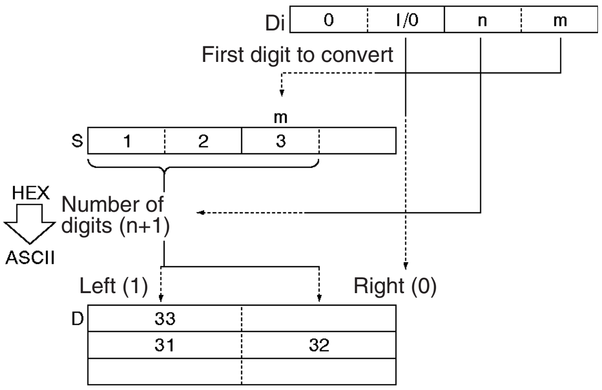
Flags

Name	Label	Operation
Error Flag	ER	• ON if the content of Di is not within the specified ranges. • OFF in all other cases.

Function

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.

A parity specification (bits 12 to 15 of K) is possible in the leftmost bit of the ASCII code data, and this can be converted to an odd or even parity bit (the number of bits that are 1 of the eight bits is adjusted to odd or even).



Hint

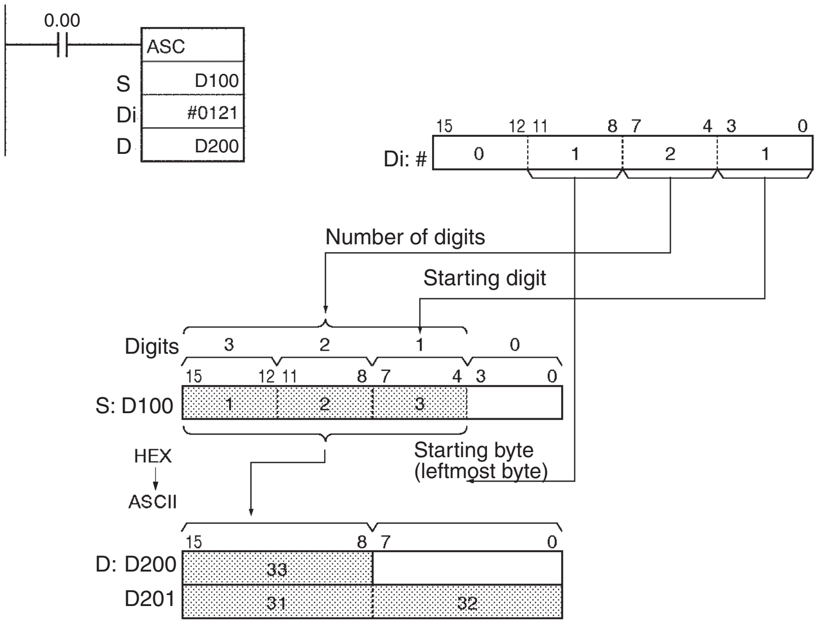
- The parity bit is appended to the data to enable detection of errors when the data is transmitted. By adding this bit, the number of bits that are 1 in the data can be indicated as odd or even, and if the number of 1s in the received data is not similarly odd or even, it is assumed that an error has occurred.
- With CS1-H CPU Units, CJ1-H CPU Units, CJ1M CPU Units, CS1D CPU Units with unit version 4.0 or later and CJ2 CPU Units, there are instructions to convert 4, 8, and 16 digits of numeric data to ASCII (STR4(524), STR8(527), and STR16(528)).

Precaution

- When multiple digits are specified in the number of digits to be converted (K), the digits are converted in order from the starting conversion digit going left (returns to digit 0 after digit 3), and the conversion results are stored in order from the output position of D going to the left word side (in units of 8 bits).
- Among the data in the conversion result output word, data in positions that are not to be output are held.
- When converting multiple digits, take care that D+2 and D+2CH do not exceed the area.

Example Programming

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) = leftmost byte, the number of digits to read = 3, and the starting digit (when reading) = digit 1.



● Example of ASCII code conversion

Content of conversion data digits					Conversion output data								
Value	Bit content				Code	(MSB) bit content (LSB)							
0	0	0	0	0	#30	*	0	1	1	0	0	0	0
1	0	0	0	1	#31	*	0	1	1	0	0	0	1
2	0	0	1	0	#32	*	0	1	1	0	0	1	0
3	0	0	1	1	#33	*	0	1	1	0	0	1	1
4	0	1	0	0	#34	*	0	1	1	0	1	0	0
5	0	1	0	1	#35	*	0	1	1	0	1	0	1
6	0	1	1	0	#36	*	0	1	1	0	1	1	0
7	0	1	1	1	#37	*	0	1	1	0	1	1	1
8	1	0	0	0	#38	*	0	1	1	1	0	0	0
9	1	0	0	1	#39	*	0	1	1	1	0	0	1
A	1	0	1	0	#41	*	1	0	0	0	0	0	1
B	1	0	1	1	#42	*	1	0	0	0	0	1	0
C	1	1	0	0	#43	*	1	0	0	0	0	1	1
D	1	1	0	1	#44	*	1	0	0	0	1	0	0
E	1	1	1	0	#45	*	1	0	0	0	1	0	1
F	1	1	1	1	#46	*	1	0	0	0	1	1	0

* Parity bit - changes according to the parity specification.

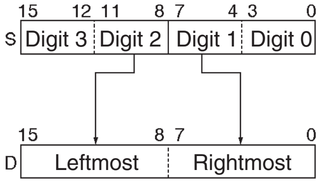
Parity

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.

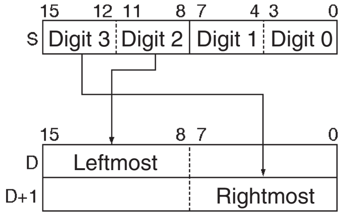
- When no parity (0) is designated, the leftmost bit will always be zero. When even parity (1) is designated, the leftmost bit will be adjusted so that the total number of ON bits is even. When odd parity (2) is designated, the leftmost bit of each ASCII character will be adjusted so that there is an 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

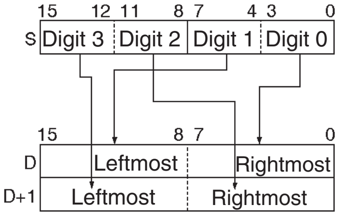
Di: #0011



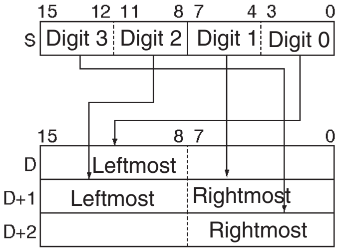
Di: #0112



Di: #0030

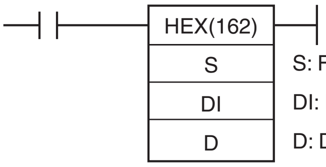


Di: #0130



HEX

Instruction	Mnemonic	Variations	Function code	Function
ASCII TO HEX	HEX	@HEX	162	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.

	HEX
Symbol	 S: First source word DI: Digit designator D: Destination word

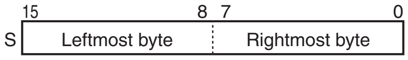
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

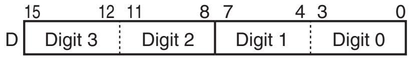
Operand	Description	Data type	Size
S	First source word	UINT	Variable
DI	Digit designator	UINT	1
D	Destination word	UINT	1

S: First Source Word



The conversion start byte and bytes to the left are treated as ASCII code and converted to hex (returns to the rightmost byte after the leftmost byte.

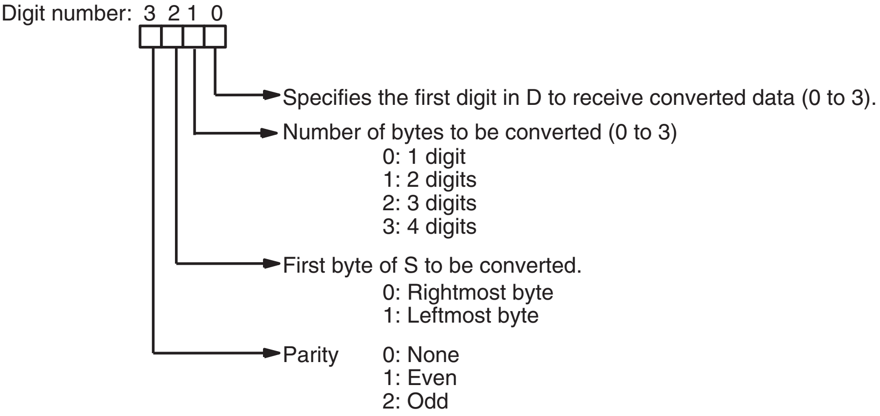
D: Destination Word



The results of conversion to hex are stored from the starting digit going left (returns to digit 0 after digit 3)

DI: Digit Designator

The digit designator specifies various parameters for the conversion, as shown in the following diagram.



Specifies the first digit in D to receive converted data (0 to 3).

Number of bytes to be converted (0 to 3)
0: 1 digit
1: 2 digits
2: 3 digits
3: 4 digits

First byte of S to be converted.
0: Rightmost byte
1: Leftmost byte

Parity
0: None
1: Even
2: Odd

● Operand Specifications

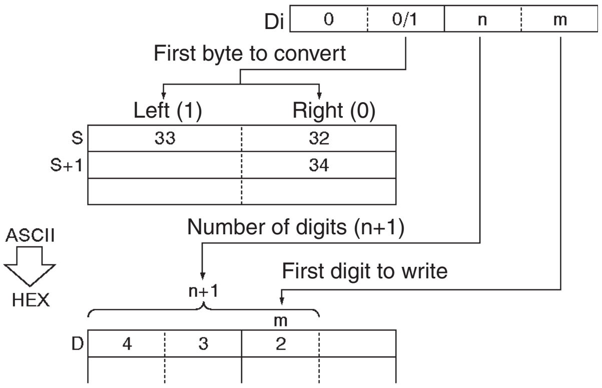
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
DI											OK	OK						
D											---	---						

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.

Function

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. When converting data, the leftmost bit of the ASCII code data can be treated as an odd or even parity bit according to the parity specification.



Hint

- The parity bit is appended to the data to enable detection of errors when the data is transmitted. By adding this bit, the number of bits that are 1 in the data can be indicated as odd or even, and if the number of 1s in the received data is not similarly odd or even, it is assumed that an error has occurred.
- With CS1-H CPU Units, CJ1-H CPU Units, CJ1M CPU Units, CS1D CPU Units with unit version 4.0 or later and CJ2 CPU Units, there are instructions to convert ASCII to 4, 8, and 16 digits of numeric data (NUM4(517), NUM8(520), and NUM16(522)).

Precaution

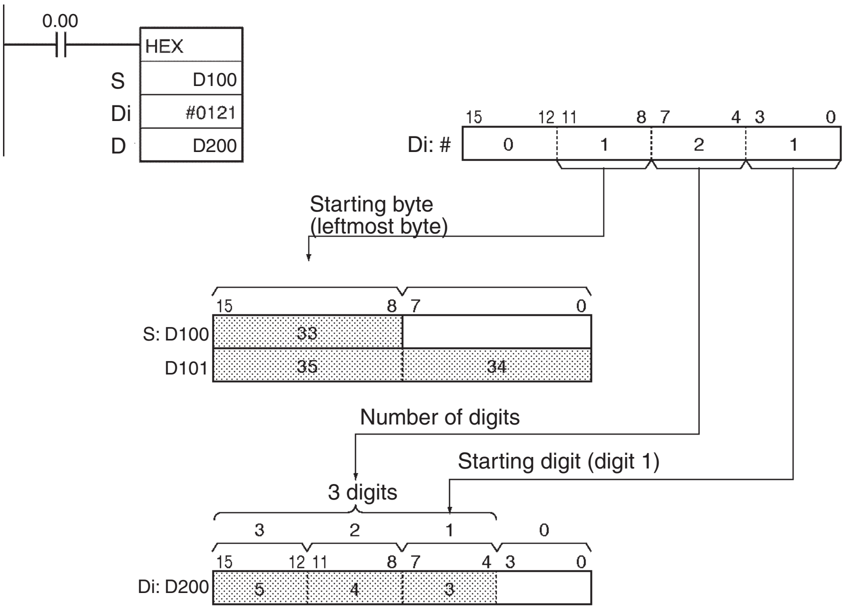
- When multiple digits are specified in the number of digits to be converted (C), the digits are converted in order from the starting conversion position (C) of S going to the left word side, and the conversion results are stored in order from the output starting bit (C) of D going to the left (returns to digit 0 after digit 3).
- Among the data in the conversion result output word, data of bits that are not to be output are held (kept the same as before).
- The following table shows ASCII data which can be contained in the source word(s) (excluding parity bits) and corresponding hexadecimal digits.

ASCII data (2 hexadecimal digits)	Hexadecimal digits
30 to 39	0 to 9
41 to 46	A to F

Example Programming

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) = leftmost byte, the number of bytes to read = 3, and the starting digit (when writing) = digit 1.)

HEX(162) converts three bytes of ASCII data (3 characters) beginning with the leftmost byte of D100 into their hexadecimal equivalents and writes this data to D200 beginning with digit 1.



● 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.

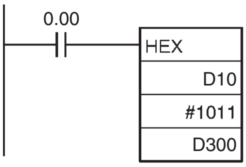
Output example

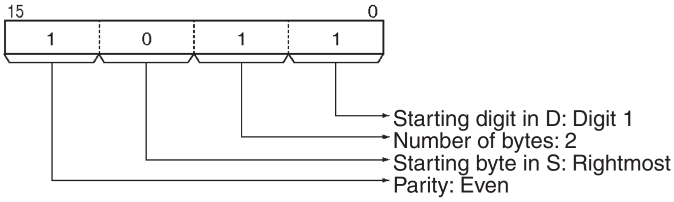
Conversion data									Output result (hex data)				
ASCII code	(MSB) bit content (LSB)								Value	Bit content			
#30	*	0	1	1	0	0	0	0	0	0	0	0	0
#31	*	0	1	1	0	0	0	1	1	0	0	0	1
#32	*	0	1	1	0	0	1	0	2	0	0	1	0
#33	*	0	1	1	0	0	1	1	3	0	0	1	1
#34	*	0	1	1	0	1	0	0	4	0	1	0	0
#35	*	0	1	1	0	1	0	1	5	0	1	0	1
#36	*	0	1	1	0	1	1	0	6	0	1	1	0
#37	*	0	1	1	0	1	1	1	7	0	1	1	1
#38	*	0	1	1	1	0	0	0	8	1	0	0	0
#39	*	0	1	1	1	0	0	1	9	1	0	0	1
#41	*	1	0	0	0	0	0	1	A	1	0	1	0
#42	*	1	0	0	0	0	1	0	B	1	0	1	1
#43	*	1	0	0	0	0	1	1	C	1	1	0	0
#44	*	1	0	0	0	1	0	0	D	1	1	0	1
#45	*	1	0	0	0	1	0	1	E	1	1	1	0
#46	*	1	0	0	0	1	1	0	F	1	1	1	1

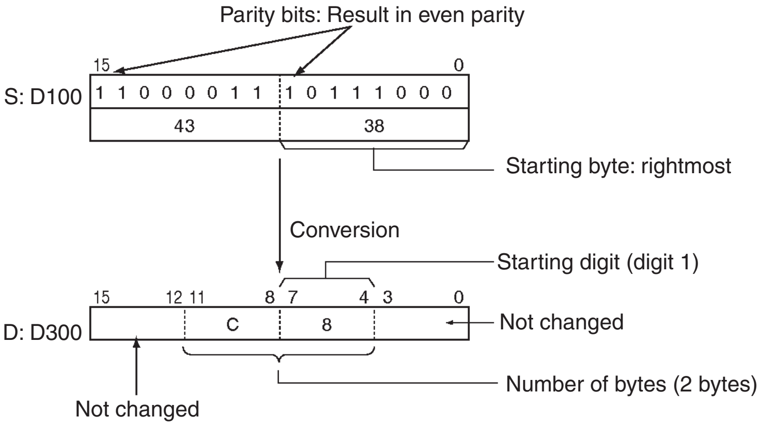
* Parity bit - changes according to the parity specification.

When CIO 0.00 is ON in the following example, HEX(162) converts the ASCII data in D10 beginning with the rightmost byte and writes the hexadecimal equivalents in D300 beginning with digit 1.

The digit designator setting of #1011 specifies even parity, the starting byte (when reading) = rightmost byte, the number of bytes to read = 2, and the starting digit (when writing) = digit 1.)

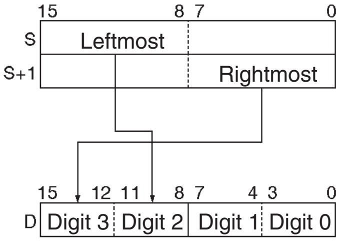




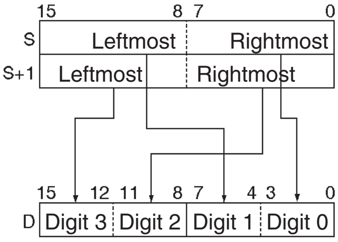


- Example of converting multiple bytes of ASCII code to hex

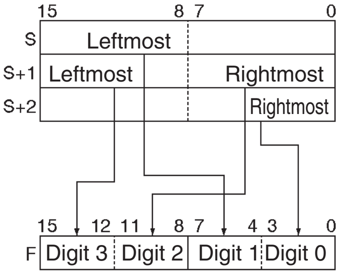
Di: #0112



Di: #0030

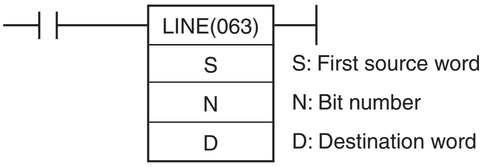


Di: #0131



LINE

Instruction	Mnemonic	Variations	Function code	Function
COLUMN TO LINE	LINE	@LINE	063	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.

	LINE
Symbol	

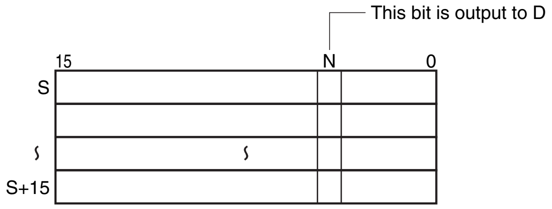
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	WORD	16
N	Bit number	UINT	1
D	Destination word	UINT	1

S: First Source Word

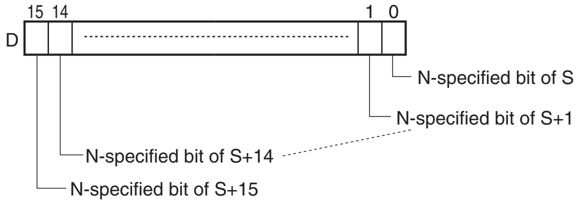


Note S and S+15 must be in the same data area.

N: Bit Number

Specifies the bit number (0000 to 000F or &0 to &15) to be copied from the source words.

D: Destination Word



● Operand Specifications

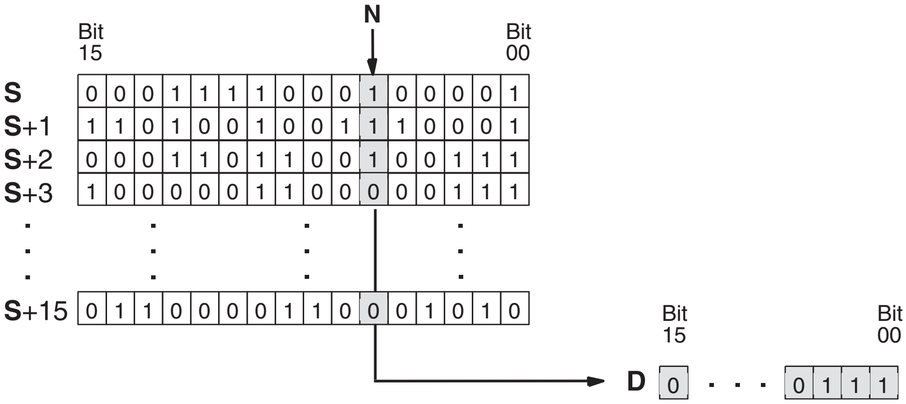
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
N											OK	OK						
D											---							

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.

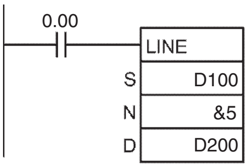
Function

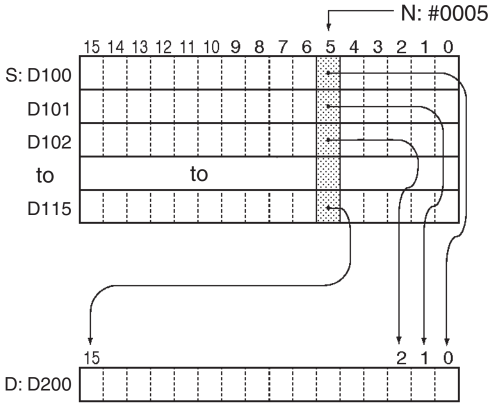
LINE(063) copies the 16 bits with bit number N from the 16-word range S to S+15 to the destination word D. Bit N of S+m is copied to bit m of D, i.e., bit N of S is copied to bit 00 of D and bit N of S+15 is copied to bit 15 of D.



Example Programming

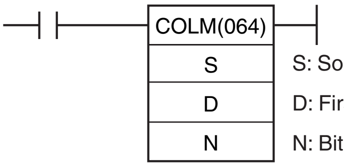
When CIO 0.00 is ON in the following example, LINE(063) copies bit 5 from D100 to D115 to the 16 bits in D200.





COLM

Instruction	Mnemonic	Variations	Function code	Function
LINE TO COLUMN	COLM	@COLM	064	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).

	COLM
Symbol	 S: Source word D: First destination word N: Bit number

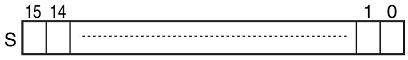
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

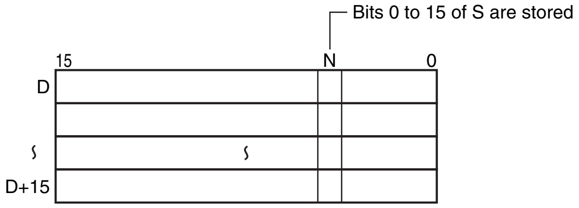
Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
D	First destination word	WORD	16
N	Bit number	UINT	1

S: Source word



D: First Destination Word



N-specified bit of D: Bit 0 of S is output

N-specified bit of D+1: Bit 1 of S is output

:
:
:

N-specified bit of D+15: Bit 15 of S is output

N: Bit Number

Specifies the bit number (0000 to 000F or &0 to &15) to be overwritten by the source word.

Note D and D+15 must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
D											---	---						
N											OK	OK						

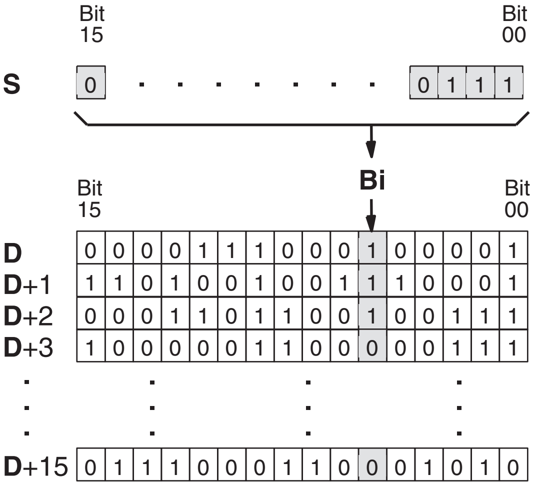
Flags

Name	Label	Operation
Error Flag	ER	• ON if N is not within the specified range of 0000 to 000F. • OFF in all other cases.
Equal Flag	=	• ON if bit N is 0 in all 16 words D to D+15 after execution. • OFF in all other cases.

Function

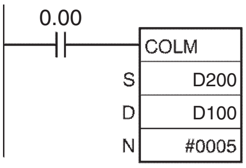
COLM(064) copies the 16 bits from S to the 16 bits with bit number N in the 16-word range D to D+15. Bit m of S is copied to bit N of D+m, i.e., bit 00 of S is copied to bit N of D and bit 15 of S is copied to bit N of D+15.

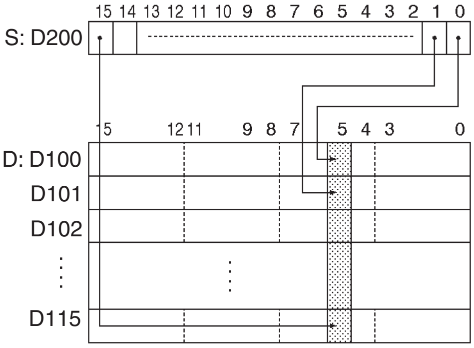
Data other than the specified bits of the conversion result output word are held (kept the same as before).



Example Programming

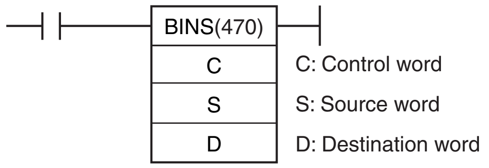
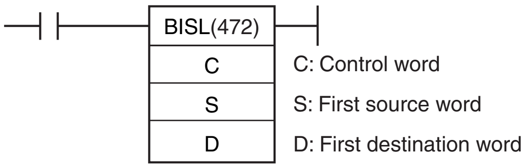
When CIO 0.00 is ON in the following example, COLM(064) copies the 16 bits in D200 (bits 00 through 15) to bit 5 in D100 through D115.





BINS/BISL

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BCD TO BINARY	BINS	@BINS	470	Converts one word of signed BCD data to one word of signed binary data.
DOUBLE SIGNED BCD TO BINARY	BISL	@BISL	472	Converts double signed BCD data to double signed binary data.

	BINS	BISL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		BINS	BISL	BINS	BISL
C	Control word	UINT	UINT	1	1
S	BINS: Source word BISL: First source word	WORD	DWORD	1	2
D	BINS: Destination word BISL: First destination word	INT	DINT	1	2

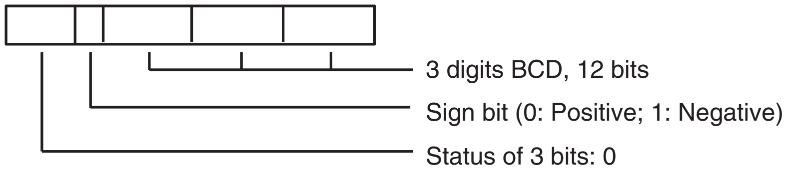
C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

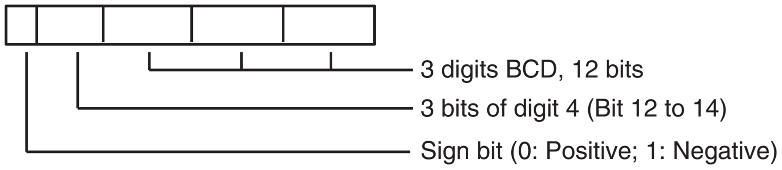
S: Source word

BINS

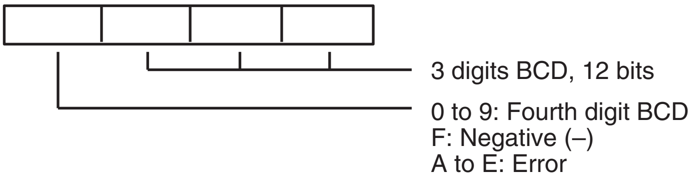
- C = 0 (Input Data Range: -999 to 999 BCD)



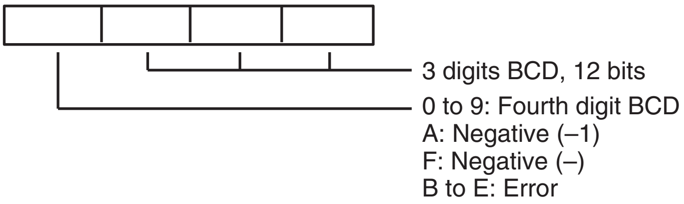
- C = 1 (Input Data Range: -7999 to 7999 BCD)



- C = 2 (Input Data Range: -999 to 9999 BCD)

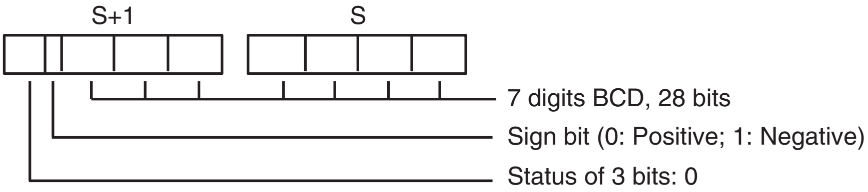


- C = 3 (Input Data Range: -1999 to 9999 BCD)

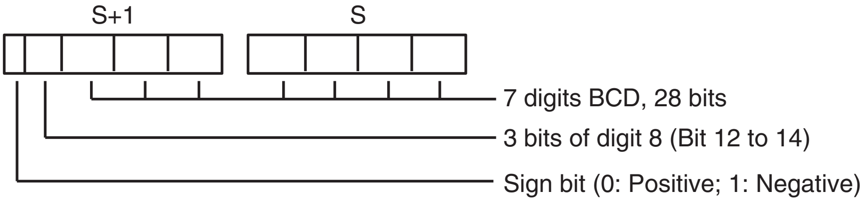


BISL

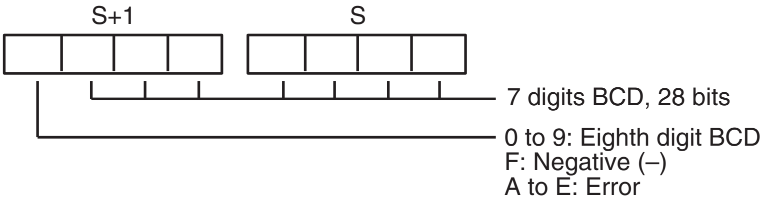
- C = 0 (Input Data Range: -999 9999 to 999 9999 BCD)



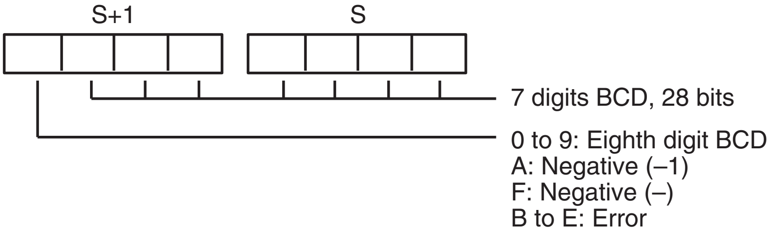
- C = 1 (Input Data Range: -7999 9999 to 7999 9999 BCD)



- C = 2 (Input Data Range: -999 9999 to 9999 9999 BCD)



- C = 3 (Input Data Range: -1999 9999 to 9999 9999 BCD)



Operand Specifications

BINS

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---							
D																		

BISL

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
D																		

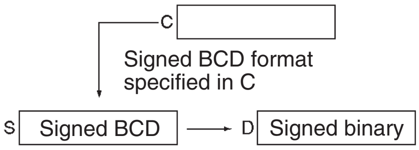
Flags

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/S+1 is A to E. • ON if C=0003 and the leftmost digit of S/S+1 is B to E. • ON if the content of S/S+1 and S is not BCD. • OFF in all other cases.
Equal Flag	=	• ON if D/D+1 contains 0000 0000 after execution. • OFF in all other cases.
Negative Flag	N	• ON if bit 15 of D/D+1 is ON after execution. • OFF in all other cases.

Function

● BINS

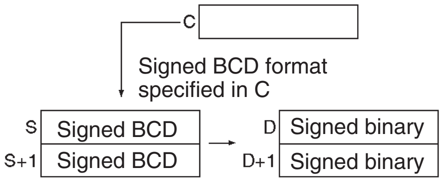
BINS(470) converts the signed BCD data in S to signed binary data and writes the result in according to the format setting in the control word (C).



Setting	Signed BCD values	Signed binary values
C=0	–999 to –1 and 0 to 999	FC19 to FFFF and 0000 to 03E7
C=1	–7999 to –1 and 0 to 7999	E0C1 to FFFF and 0000 to 1F3F
C=2	–999 to –1 and 0 to 9999	FC19 to FFFF and 0000 to 270F
C=3	–1999 to –1 and 0 to 9999	F831 to FFFF and 0000 to 270F

● BISL

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 according to the format setting in the control ward (C).



Setting	Signed BCD values	Signed binary values
C=0	–999 9999 to –1	FF67 6981 to FFFF FFFF
	0 to 999 9999	0000 0000 to 0098 967F
C=1	–7999 9999 to –1	FB3B 4C01 to FFFF FFFF
	0 to 7999 9999	0000 0000 to 04C4 B3FF
C=2	–999 9999 to –1	FF67 6981 to FFFF FFFF
	0 to 9999 9999	0000 0000 to 05F5 E0FF
C=3	–1999 9999 to –1	FECE D301 to FFFF FFFF
	0 to 9999 9999	0000 0000 to 05F5 E0FF

Hint

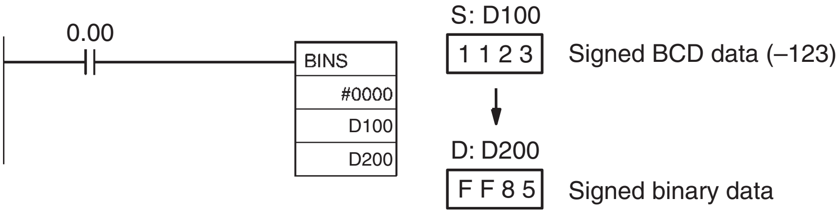
Some Special I/O Units output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data with BINS(470)/B/SL(472).

Example Programming

● BINS

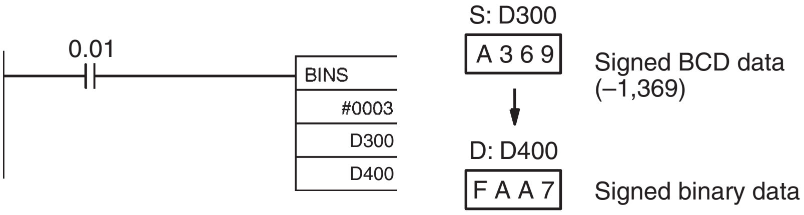
BCD Format 0 (C=#0000)

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 3 (C=#0003)

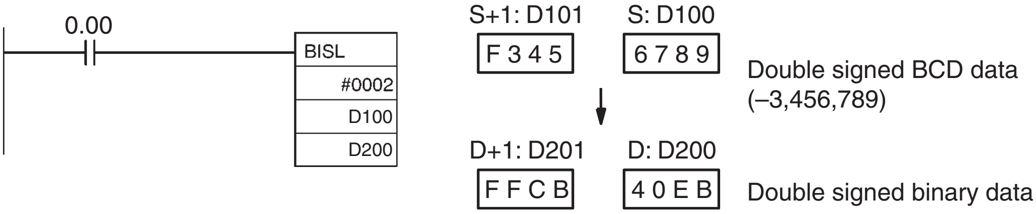
When CIO 0.01 is ON in the following example, the signed BCD data format and range in D100 are checked against the format specified in the control word (0003). The source data is correct, so the signed BCD data in D300 is converted to signed binary and output to D400.



● BISL

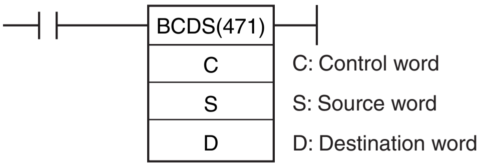
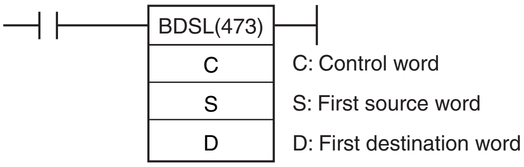
BCD Format 2 (C=#0002)

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 (2). 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.



BCDS/BDSL

Instruction	Mnemonic	Variations	Function code	Function
SIGNED BINARY TO BCD	BCDS	@BCDS	471	Converts one word of signed binary data to one word of signed BCD data.
DOUBLE SIGNED BINARY TO BCD	BDSL	@BDSL	473	Converts double signed binary data to double signed BCD data.

	BCDS	BDSL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		BCDS	BDSL	BCDS	BDSL
C	Control word	UINT	UINT	1	1
S	BCDS: Source word BDSL: First source word	INT	DINT	1	2
D	BCDS: Destination word BDSL: First destination word	WORD	DWORD	1	2

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

S: Source Word

BCDS

Setting	Allowed values for S
C=0	FC19 to FFFF or 0000 to 03E7
C=1	E0C1 to FFFF or 0000 to 1F3F
C=2	FC19 to FFFF or 0000 to 270F
C=3	F831 to FFFF or 0000 to 270F

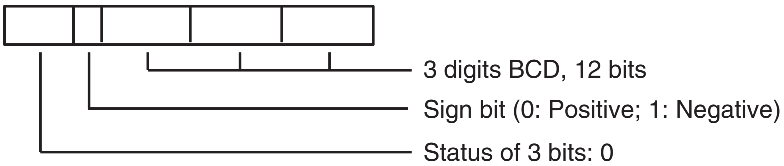
BDSL

Setting	Allowed values for S+1 and S
C=0	FF67 6981 to FFFF FFFF or 0000 0000 to 0098 967F
C=1	FB3B 4C01 to FFFF FFFF or 0000 0000 to 04C4 B3FF
C=2	FF67 6981 to FFFF FFFF or 0000 0000 to 05F5 E0FF
C=3	FECE D301 to FFFF FFFF or 0000 0000 to 05F5 E0FF

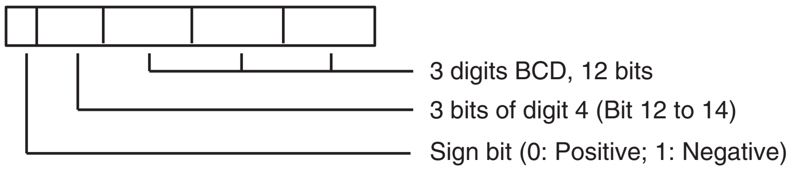
D: Destination Word

BCDS

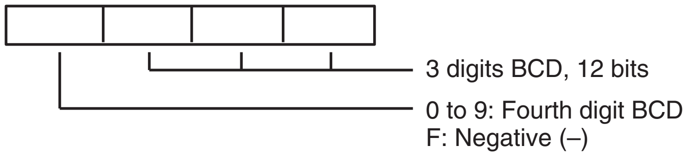
- C = 0 (Output Data Range: -999 to 999 BCD)



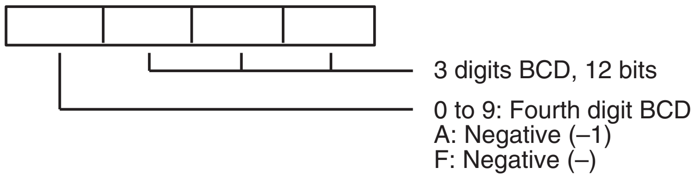
- C = 1 (Output Data Range: -7999 to 7999 BCD)



- C = 2 (Output Data Range: -999 to 9999 BCD)

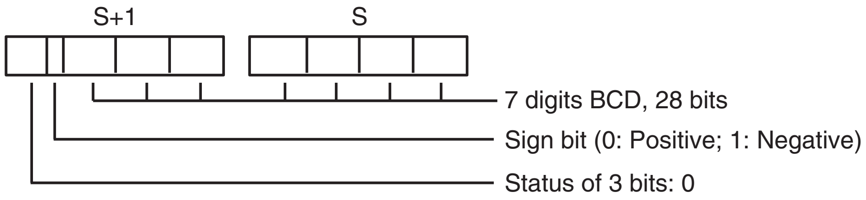


- C = 3 (Output Data Range: -1999 to 9999 BCD)

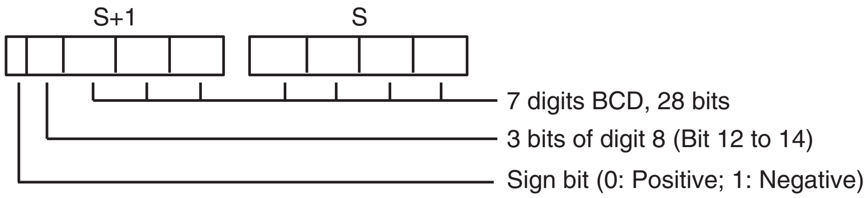


BDSL

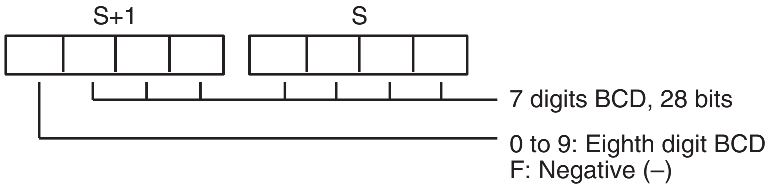
- C = 0 (Output Data Range: -999 9999 to 999 9999 BCD)



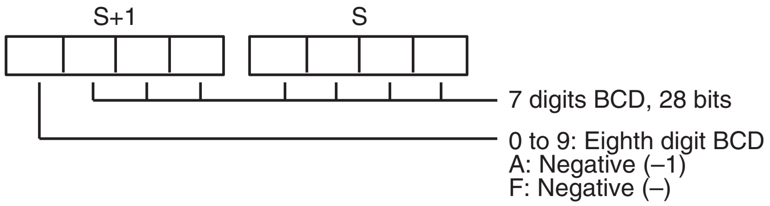
- C = 1 (Output Data Range: -7999 9999 to 7999 9999 BCD)



- C = 2 (Output Data Range: -999 9999 to 9999 9999 BCD)



- C = 3 (Output Data Range: -1999 9999 to 9999 9999 BCD)



● Operand Specifications

BCDS

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---							
D																		

BDSL

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
D																		

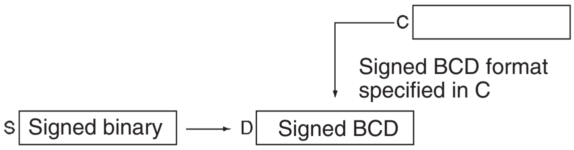
Flags

Name	Label	Operation	
Error Flag	ER	• ON if C is not within the specified range of 0000 to 0003. • ON if C=0 and the source data is not within the allowed ranges (FC19 to FFFF or 0000 to 03E7). • ON if C=1 and the source data is not within the allowed ranges (E0C1 to FFFF or 0000 to 1F3F). • ON if C=2 and the source data is not within the allowed ranges (FC19 to FFFF or 0000 to 270F). • ON if C=3 and the source data is not within the allowed ranges (F831 to FFFF or 0000 to 270F). • OFF in all other cases.	• ON if C is not within the specified range of 0000 to 0003. • ON if C=0 and the source data is not within the range: • FF67 6981 to FFFF FFFF or 0000 0000 to 0098 967F. • ON if C=1 and the source data is not within the range: • FB3B 4C01 to FFFF FFFF or 0000 0000 to 04C4 B3FF. • ON if C=2 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.
Equal Flag	=	• ON if D is 0000 after execution. • OFF in all other cases.	
Negative Flag	N	• ON if C=0 or 1 and the result's sign bit is ON after execution. • ON if C=2 and the leftmost digit of the result is F. • ON if C=3 and the leftmost digit of the result is A or F. • OFF in all other cases.	

Function

● BCDS

BCDS(471) converts the signed binary data in S to signed BCD data and writes the result in D according to the format setting in the control word (C).

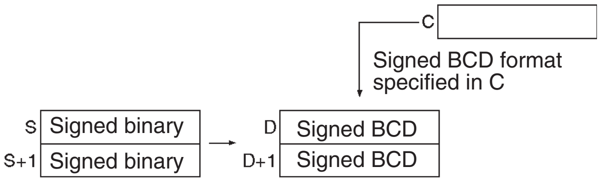


The following table shows the possible signed binary values for each signed BCD format. An error will occur if the source data is not within the allowed range for the specified signed BCD format.

Setting	Signed binary values	Signed BCD values
C=0	FC19 to FFFF and 0000 to 03E7	–999 to –1 and 0 to 999
C=1	E0C1 to FFFF and 0000 to 1F3F	–7999 to –1 and 0 to 7999
C=2	FC19 to FFFF and 0000 to 270F	–999 to –1 and 0 to 9999
C=3	F831 to FFFF and 0000 to 270F	–1999 to –1 and 0 to 9999

- **BDSL**

BDSL(473) converts the double singled binary data in S+1 and S to signed BCD data and writes the result in D+1 and D according to the format setting in the control word (C).



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	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

Precautions

Values of -0 in the source data will be treated as 0 and will not cause an error.

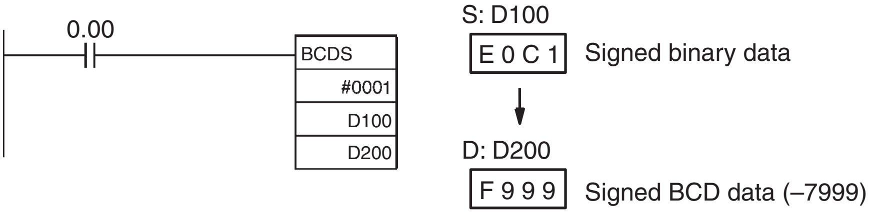
Hint

Some Special I/O Units require signed BCD data inputs. BCDS(471)/BDSL(473) can be used to convert signed binary data for output to these Units.

Example Programming

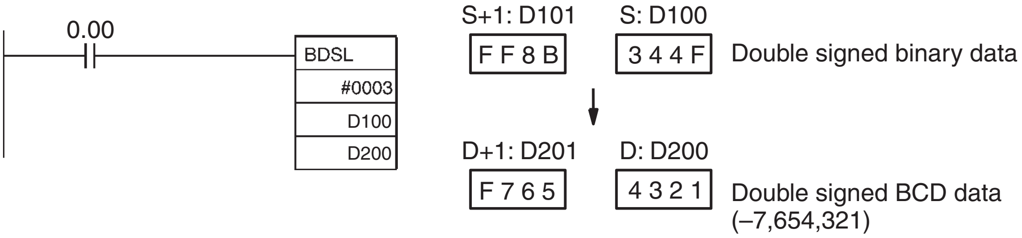
- **BCDS**

When 0.00 is ON, it is first checked if the data range of D100 is based on the data format specification 0001. Here the check result is okay, so the signed binary data in data memory D100 is converted to signed BCD and output to D200.



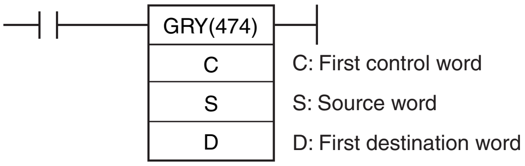
- **BDSL**

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.



GRY

Instruction	Mnemonic	Variations	Function code	Function
GRAY CODE CONVERT	GRY	@GRY	474	Converts the gray binary code in a specified word to standard binary data, BCD data, or an angle at the specified resolution.

	GRY
Symbol	

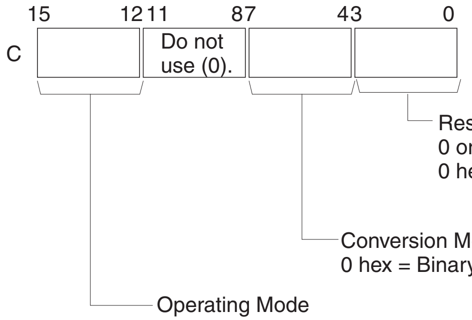
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	---	3
S	Source word	---	1
D	First destination word	---	2

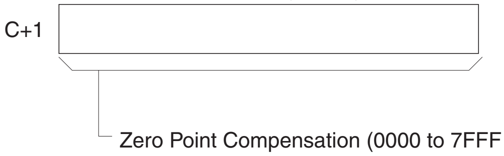
C: Control Word



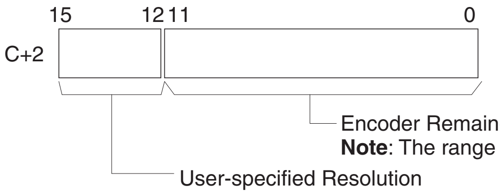
Resolution
0 or 1 to F hex (1 to 15 decimal) bits
0 hex = User specified in bits 12 to 15 of C+2.

Conversion Mode
0 hex = Binary Mode, 1 hex = BCD Mode, 2 hex = 360° Mode

Operating Mode
0 hex = Gray binary code conversion



Zero Point Compensation (0000 to 7FFF Hex (Binary Data))
Note: Zero point compensation that exceeds the resolution set in the word C of the control data cannot be specified.



Encoder Remainder Compensation (Binary Data)
Note: The range that can be set depends on the user-specified resolution.

User-specified Resolution
0 hex = 256, 1 hex = 360, 2 hex = 720, 3 hex = 1,024, 4 to F hex = Do not use.

Note: The above setting is valid when the resolution is set to 0 hex in bits 00 to 03 of C.

S: Source Word

S	

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

D	Rightmost word
D+1	Leftmost 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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S											OK	OK						
D											---	---						

Flags

Name	Label	Operation
Error Flag	P_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.
Equal Flag	P_EQ	OFF in all cases.
Negative Flag	P_N	OFF in all cases.

Function

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.

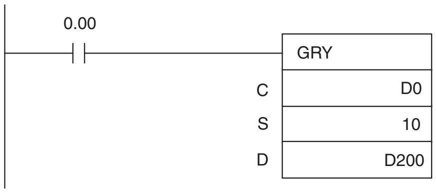
Hint

GRY(474) is normally used when inputting, through a DC Input Unit, a parallel signal (2n) from an absolute encoder that outputs a gray binary code.

Note 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.

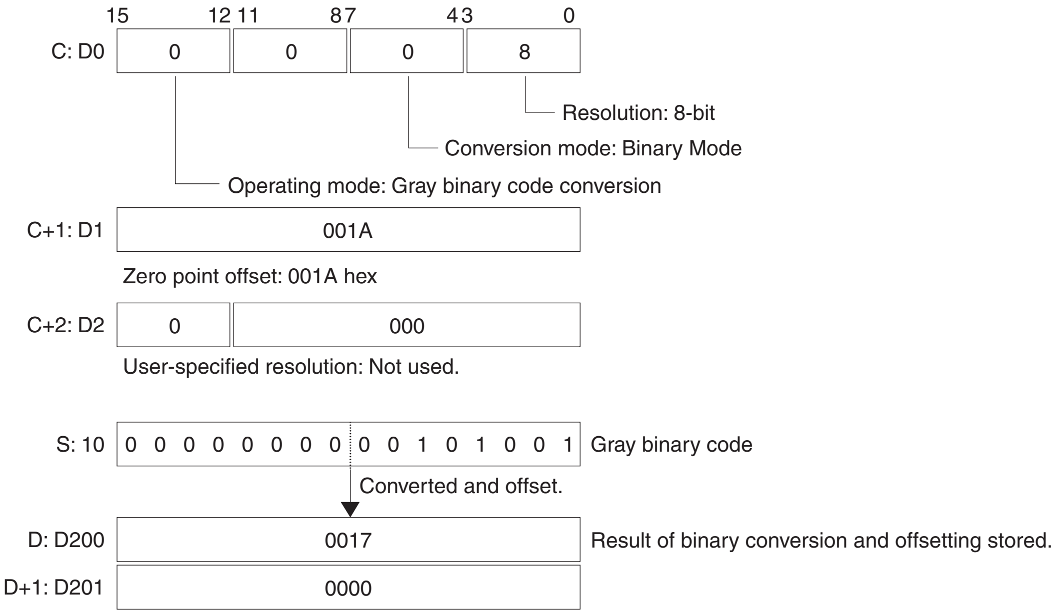
Example Programming

When CIO 0.00 is ON in the following example, the gray binary code in CIO 10 is converted according to the settings in the control data in D0 to D2 and the result is output to D200 and D201.



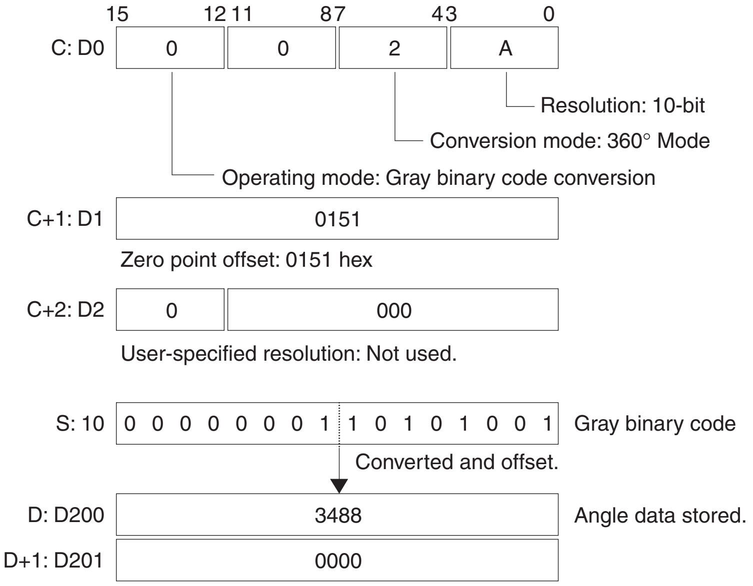
Example 1:

Converting to Binary Data with an 8-bit Resolution and Zero Point Offset of 001A Hex



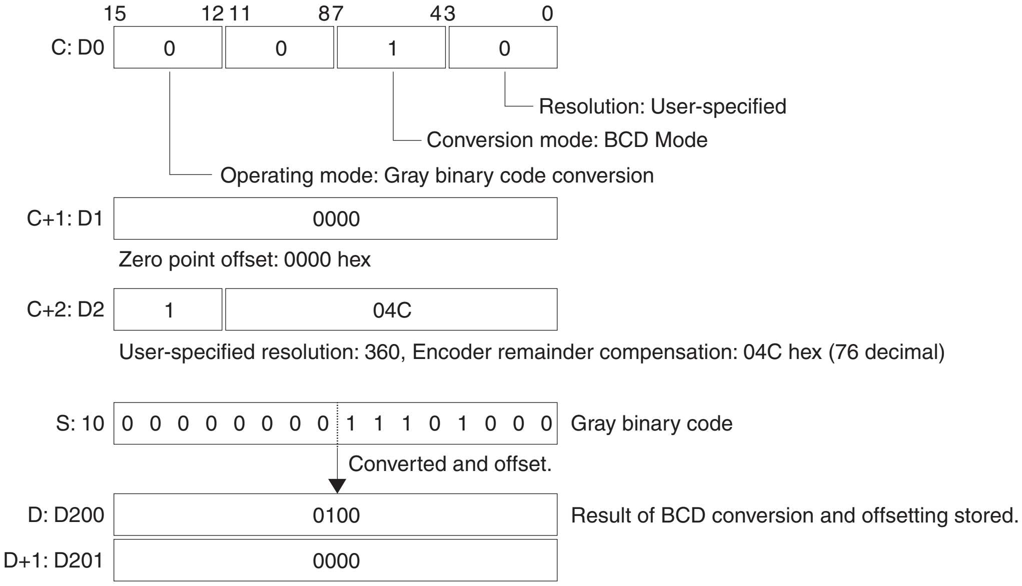
Example 2:

Converting to Angle Data with a 10-bit Resolution and Zero Point Offset of 0151 Hex



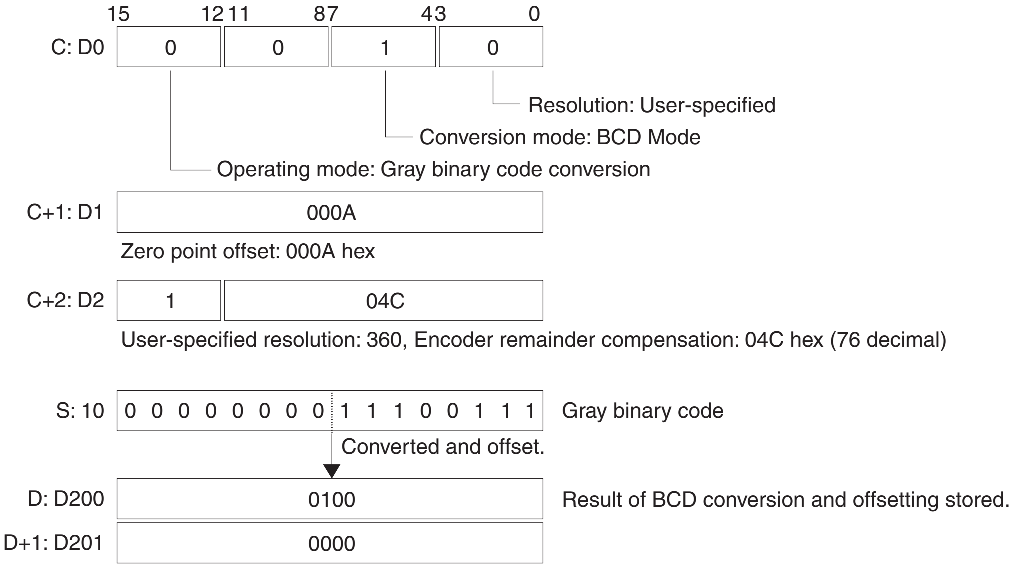
Example 3:

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



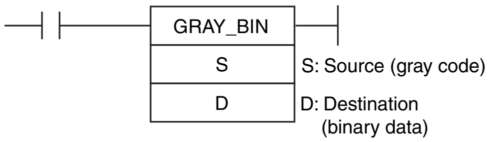
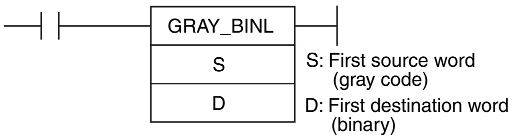
Example 4:

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



GRAY_BIN/GRAY_BINL

Instruction	Mnemonic	Variations	Function code	Function
GRAY CODE TO BINARY CONVERT	GRAY_BIN	@GRAY_BIN	478	Converts the specified word of gray code to one word of binary data.
DOUBLE GRAY CODE TO BINARY CONVERT	GRAY_BINL	@GRAY_BINL	479	Converts the specified two words of gray code to two words of binary data.

	GRAY_BIN	GRAY_BINL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		GRAY_BIN	GRAY_BINL	GRAY_BIN	GRAY_BINL
S	Gray code data	WORD	DWORD	1	2
D	Binary data	WORD	DWORD	1	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
D											---							

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the results of conversion is negative (i.e., if the gray code data is #8000 to #FFFF). • OFF in all other cases.
Equal Flag	P_EQ	Unchanged.
Negative Flag	P_N	Unchanged.

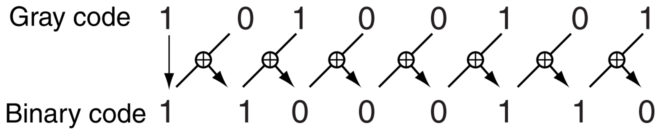
Function

● GRAY_BIN(478)

GRAY_BIN(478) converts one word of gray code to one word of binary data. S specifies the source word containing the gray code and the binary data is output to D.

● GRAY_BINL(479)

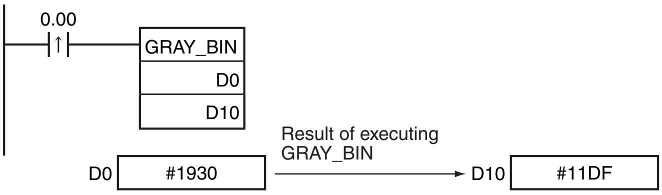
GRAY_BINL(479) converts two words of gray code to two words of binary data. S specifies the first of the two words containing the gray code and the binary data is output to D and D+1.



Example Programming

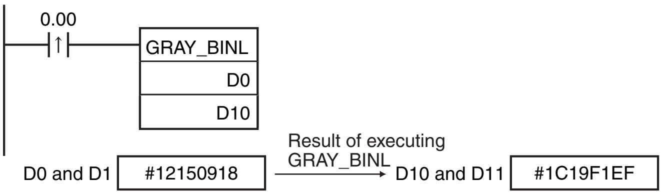
● GRAY_BIN

When CIO 0.00 turns ON in the following example, the gray code in D0 is converted to binary data and stored in D10.



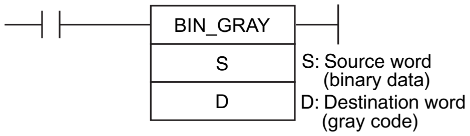
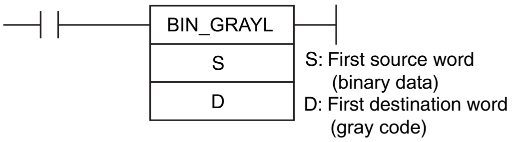
● GRAY_BINL

When CIO 0.00 turns ON in the following example, the gray code in D0 and D1 is converted to binary data and stored in D10 and D11.



BIN_GRAY/BIN_GRAYL

Instruction	Mnemonic	Variations	Function code	Function
BINARY TO GRAY CODE CONVERT	BIN_GRAY	@BIN_GRAY	480	Converts the specified word of binary data to one word of gray code.
DOUBLE BINARY TO GRAY CODE CONVERT	BIN_GRAYL	@BIN_GRAYL	481	Converts the specified two words of binary data to two words of gray code.

	BIN_GRAY	BIN_GRAYL
Symbol	 S: Source word (binary data) D: Destination word (gray code)	 S: First source word (binary data) D: First destination word (gray code)

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		BIN_GRAY	BIN_GRAYL	BIN_GRAY	BIN_GRAYL
S	Binary data	WORD	DWORD	1	2
D	Gray code data	WORD	DWORD	1	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
D											---							

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the binary code is negative (i.e., if the gray code data is #8000 to #FFFF). • OFF in all other cases.
Equal Flag	P_EQ	Unchanged.
Negative Flag	P_N	Unchanged.

Function

● BIN_GRAY(480)

BIN_GRAY(480) converts one word of binary data to one word of gray code. S specifies the source word containing the binary data and the gray code is output to D.

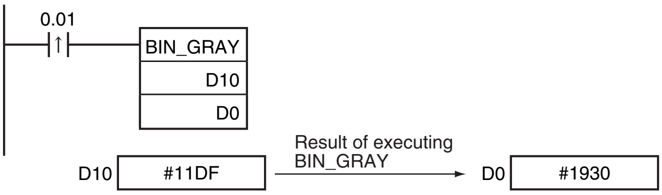
● BIN_GRAYL(481)

BIN_GRAYL(481) converts two words of binary data to two words of gray code. S specifies the first of the two words containing the binary data and the gray code is output to D and D+1.

Example Programming

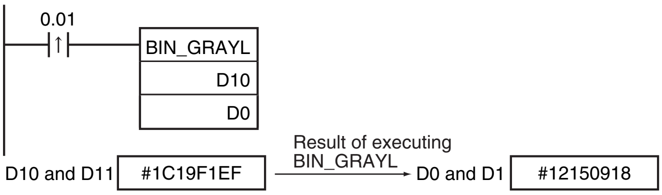
● BIN_GRAY

When CIO 0.01 turns ON in the following example, the binary data in D10 is converted to gray code and stored in D0.



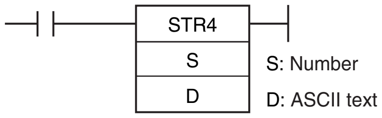
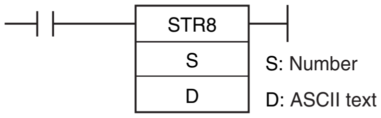
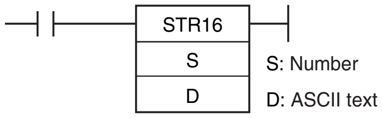
● BIN_GRAYL

When CIO 0.01 turns ON in the following example, the binary data in D10 and D11 is converted to gray code and stored in D0 and D1.



STR4/STR8/STR16

Instruction	Mnemonic	Variations	Function code	Function
FOUR-DIGIT NUMBER TO ASCII	STR4	@STR4	601	Converts a 4-digit hexadecimal number (#0000 to #FFFF) to ASCII data (4 characters).
EIGHT-DIGIT NUMBER TO ASCII	STR8	@STR8	602	Converts an 8-digit hexadecimal number (#0000 0000 to #FFFF FFFF) to ASCII data (8 characters).
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	@STR16	603	Converts a 16-digit hexadecimal number (#0000 0000 0000 0000 to #FFFF FFFF FFFF FFFF) to ASCII data (16 characters).

	STR4	STR8	STR16
Symbol			

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type			Size		
		STR4	STR8	STR16	STR4	STR8	STR16
S	Number	UINT	UDINT	ULINT	1	2	4
D	ASCII text	WORD	WORD	WORD	2	4	8

● Operand Specifications

STR4, STR8

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
D											---							

STR16

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

Flags

Name	Label	Operation
Error Flag	P_ER	OFF
Equal Flag	P_EQ	• ON if the result is 0. • OFF in all other cases.
Negative Flag	P_N	• ON when the leftmost bit of the source data is 1. • OFF in all other cases.

Function

● STR4

STR4(601) converts the numerical data in S (4-digit hexadecimal, #0000 to #FFFF) to ASCII data (4 characters) and writes the result to D and D+1.

	15 12	11 8	7 4	3 0
S	1	2	3	4

Hexadecimal: #1234

⇩

ASCII

	15 8	7 0
D	31	32
D+1	33	34

● STR8

STR8(602) converts the numerical data in S and S+1 (8-digit hexadecimal, #0000 0000 to #FFFF FFFF) to ASCII data (8 characters) and writes the result to D, D+1, D+2, and D+3.

	15 12	11 8	7 4	3 0
S	5	6	7	8
S+1	1	2	3	4

Hexadecimal: #12345678

⇩

ASCII

	15 8	7 0
D	31	32
D+1	33	34
D+2	35	36
D+3	37	38

● STR16

STR16(603) converts the numerical data in S to S+3 (16-digit hexadecimal, #0000 0000 0000 0000 to #FFFF FFFF FFFF FFFF) to ASCII data (16 characters) and writes the result to D to D+7.

	15 12	11 8	7 4	3 0
S	C	D	E	F
S+1	8	9	A	B
S+2	4	5	6	7
S+3	0	1	2	3

Hexadecimal: #1234567890ABCDEF

⇩

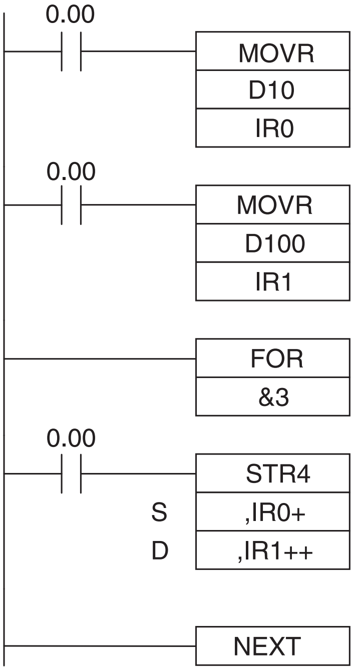
ASCII

	15 8	7 0
D	30	31
D+1	32	33
D+2	34	35
D+3	36	37
D+4	38	39
D+5	41	42
D+6	43	44
D+7	45	46

Example Programming

● Converting 3 Words of Numerical Data to ASCII Data

When CIO 0.00 is ON in the following example, the 3 words of numerical data starting at D10 are converted, one word at a time, to ASCII data. The converted ASCII data is stored in the DM Area starting at D100.



	15–12	11–8	7–4	3–0
S: D10	0	1	2	3
S+1: D11	4	5	6	7
S+2: D12	8	9	A	B

Hexadecimal

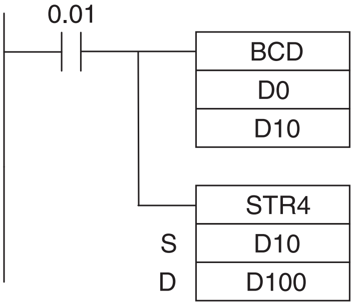
⇩

ASCII

	15–8	7–0
D: D100	30	31
D+1: D101	32	33
D+2: D102	34	35
D+3: D103	36	37
D+4: D104	38	39
D+5: D105	41	42

● Converting Decimal Data to ASCII Data in BCD Format

When CIO 0.01 is ON in the following example, the source data in D0 (&1234 in decimal) is converted to BCD data and the result is stored temporarily in D10. Next, the BCD data is converted to ASCII data and the result is output to D100 and D101.



	15–12	11–8	7–4	3–0
D0	0	4	D	2

&1234 Decimal
(#04D2 hexadecimal)

Binary (hexadecimal)

⇩

BCD

	15–12	11–8	7–4	3–0
S: D10	1	2	3	4

BCD

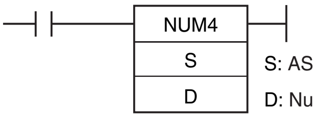
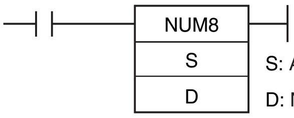
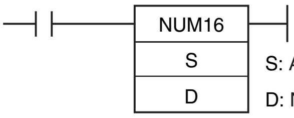
⇩

ASCII (BCD)

	15–8	7–0
D: D100	31	32
D+1: D101	33	34

NUM4/NUM8/NUM16

Instruction	Mnemonic	Variations	Function code	Function
ASCII TO FOUR-DIGIT NUMBER	NUM4	@NUM4	604	Converts 4 characters of ASCII data to a 4-digit hexadecimal number.
ASCII TO EIGHT-DIGIT NUMBER	NUM8	@NUM8	605	Converts 8 characters of ASCII data to an 8-digit hexadecimal number.
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	@NUM16	606	Converts 16 characters of ASCII data to an 16-digit hexadecimal number.

	NUM4	NUM8	NUM16
Symbol	 S: ASCII text D: Number	 S: ASCII text D: Number	 S: ASCII text D: Number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type			Size		
		NUM4	NUM8	NUM16	NUM4	NUM8	NUM16
S	ASCII text	WORD	WORD	WORD	2	4	8
D	Number	UINT	UDINT	ULINT	1	2	4

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

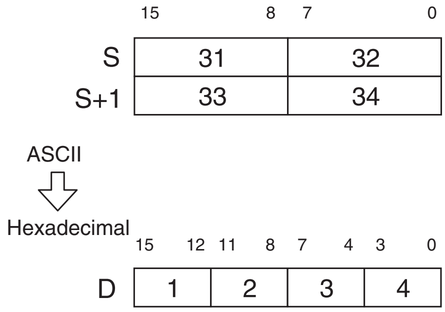
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the source words contain any ASCII characters that are not hexadecimal equivalents (0 to 9, a to f, or A to F). • OFF in all other cases.
Equal Flag	P_EQ	• ON if the result is 0. • OFF in all other cases.
Negative Flag	P_N	• ON when the leftmost bit of the source data is 1. • OFF in all other cases.

Function

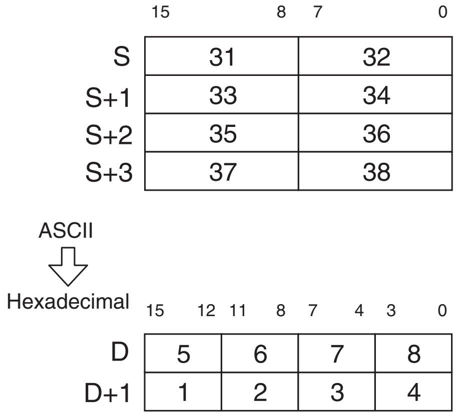
● NUM4

- NUM4(604) converts the 4 characters of ASCII data in S and S+1 to numerical data (4-digit hexadecimal) and writes the result to D.
- The Error Flag will be turned ON if the ASCII data in S and S+1 contains any characters that are not hexadecimal digits. In this case, the instruction will not be executed.



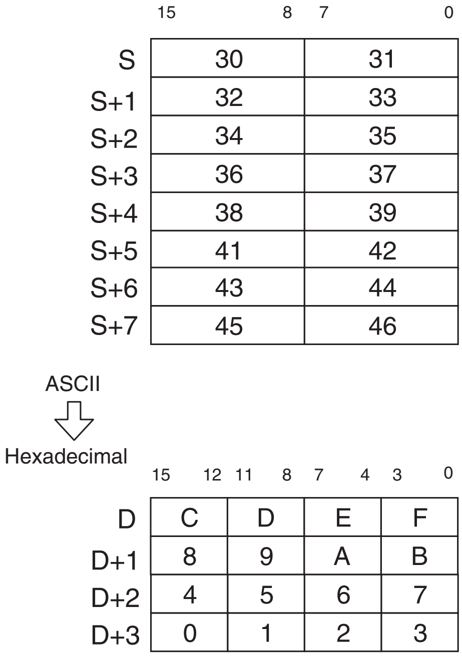
● NUM8

- NUM8(605) converts the 8 characters of ASCII data in S to S+3 to numerical data (4-digit hexadecimal) and writes the result to D and D+1.
- The Error Flag will be turned ON if the ASCII data contains any characters that are not hexadecimal digits. In this case, the instruction will not be executed.



● NUM16

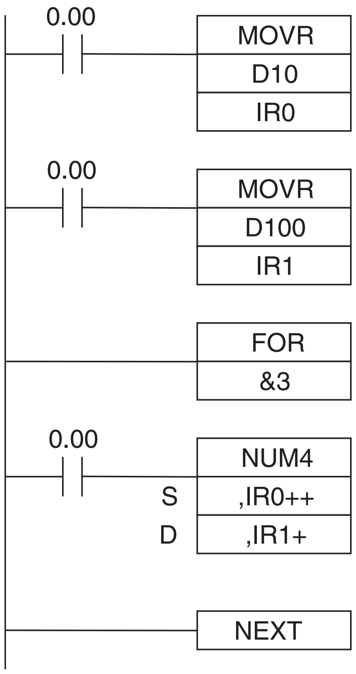
- NUM16(606) converts the 16 characters of ASCII data in S to S+7 to numerical data (4-digit hexadecimal) and writes the result to D to D+3.
- The Error Flag will be turned ON if the ASCII data contains any characters that are not hexadecimal digits. In this case, the instruction will not be executed.

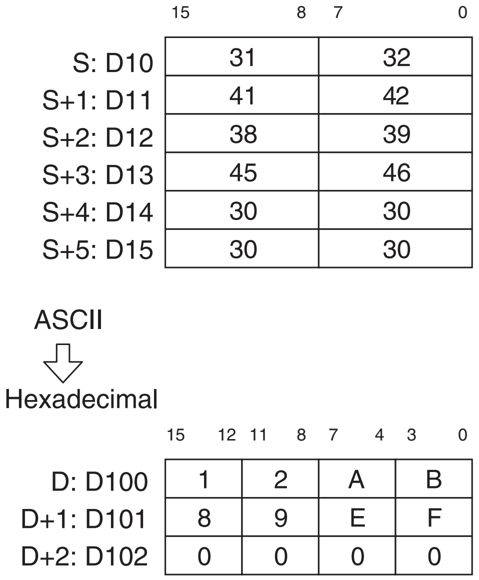


Example Programming

● Converting 3 Sets of 4 ASCII Characters to the Equivalent Hexadecimal Digits

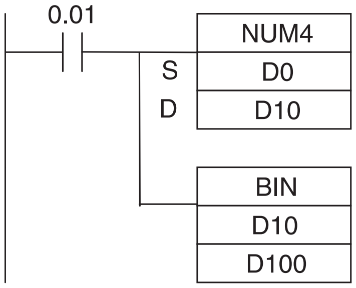
When CIO 0.00 is ON in the following example, the 6 words of ASCII data starting at D10 are converted, two words at a time, to numerical data. The converted numerical data is stored in the DM Area starting at D100.

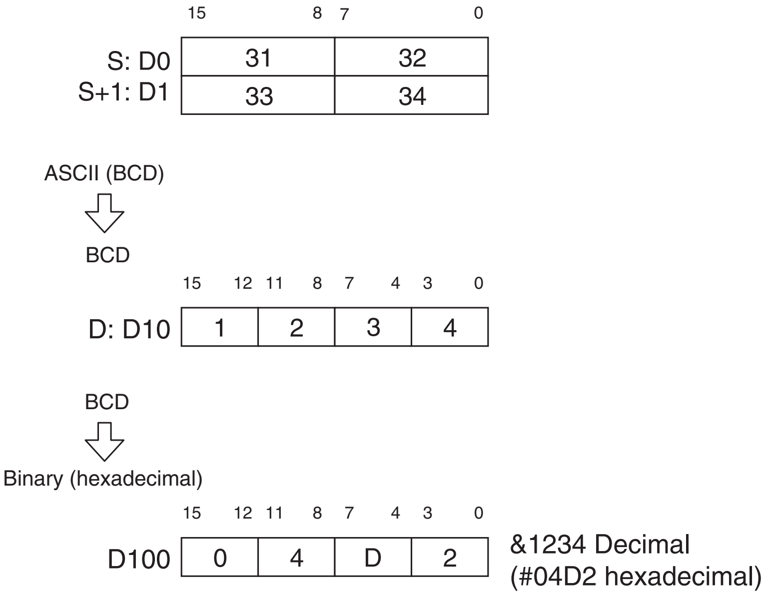




● Converting ASCII Data in BCD Format to Hexadecimal Data

When CIO 0.01 is ON in the following example, the ASCII characters in D0 and D1 are converted to BCD data and the result is stored temporarily in D10. Next, the BCD data is converted to hexadecimal and the result is output to D100.

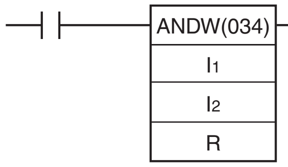
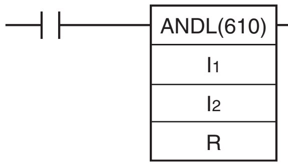




Logic Instructions

ANDW/ANDL

Instruction	Mnemonic	Variations	Function code	Function
LOGICAL AND	ANDW	@ANDW	034	Takes the logical AND of corresponding bits in single words of word data and/or constants.
DOUBLE LOGICAL AND	ANDL	@ANDL	610	Takes the logical AND of corresponding bits in double words of word data and/or constants.

	ANDW	ANDL
Symbol	 I1: Input 1 I2: Input 2 R: Result word	 I1: Input 1 I2: Input 2 R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ANDW	ANDL	ANDW	ANDL
I_1	Input 1	WORD	DWORD	1	2
I_2	Input 2	WORD	DWORD	1	2
R	Result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ANDW	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
ANDL	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

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.

Function

● ANDW

ANDW(034) takes the logical AND of data specified in I_1 and I_2 and outputs the result to R.

I_1, $I_2 \rightarrow$ R

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	0

● ANDL

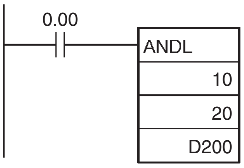
ANDL(610) takes the logical AND of data specified in I_1, I_1+1 and I_2, I_2+1 and outputs the result to R, R+1.

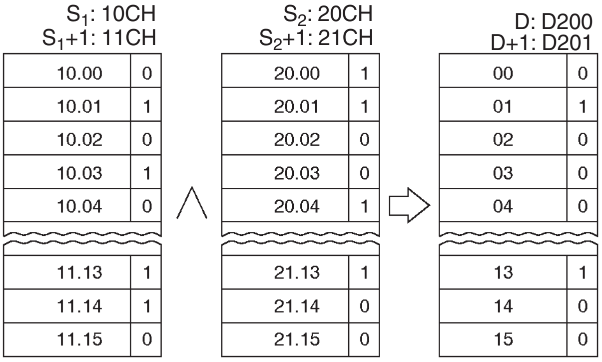
(I_1, I_1+1) • (I_2, I_2+1) $\rightarrow$ (R, R+1)

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	0
0	1	0
0	0	0

Example Programming

When the execution condition CIO 0.00 is ON, the logical AND is taken of corresponding bits in CIO 11, CIO 10 and CIO 21, CIO 20 and the results will be output to corresponding bits in D201 and D200.

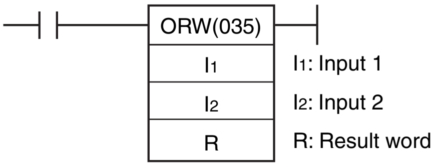
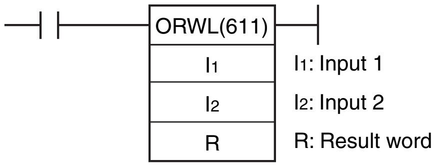




Note: The vertical arrow indicates logical AND.

ORW/ORWL

Instruction	Mnemonic	Variations	Function code	Function
LOGICAL OR	ORW	@ORW	035	Takes the logical OR of corresponding bits in single words of word data and/or constants.
DOUBLE LOGICAL OR	ORWL	@ORWL	611	Takes the logical OR of corresponding bits in double words of word data and/or constants.

	ORW	ORWL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		ORW	ORWL	ORW	ORWL
I_1	Input 1	WORD	DWORD	1	2
I_2	Input 2	WORD	DWORD	1	2
R	Result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
ORW	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
ORWL	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

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.

Function

● ORW

ORW(035) takes the logical OR of data specified in I_1 and I_2 and outputs the result to R.

$I_1, I_2 \rightarrow R$

I_1	I_2	R
1	1	1
1	0	1
0	1	1
0	0	0

● ORWL

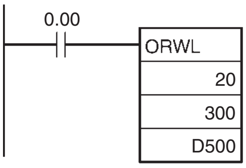
ORWL(611) takes the logical OR of data specified in I_1 and I_2 as double-word data and outputs the result to R, R+1.

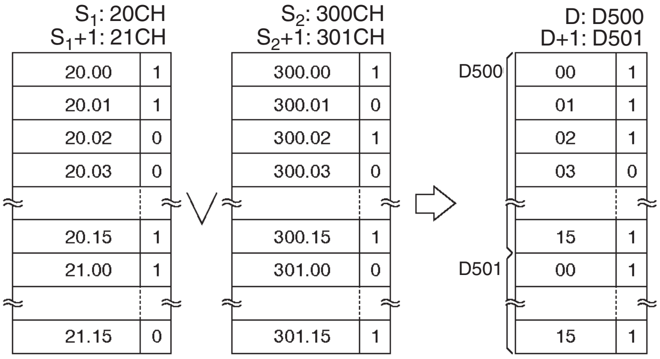
$(I_1, I_1+1) + (I_2, I_2+1) \rightarrow (R, R+1)$

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	1
0	1	1
0	0	0

Example Programming

When the execution condition CIO 0.00 is ON, the logical OR is taken of corresponding bits in CIO 21, CIO 20 and CIO 301, CIO 300 and the results will be output to corresponding bits in D501 and D500.

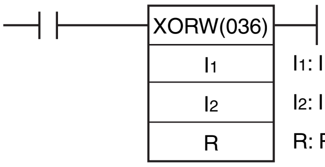
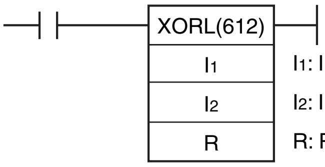




Note: The vertical arrow indicates logical OR.

XORW/XORL

Instruction	Mnemonic	Variations	Function code	Function
EXCLUSIVE OR	XORW	@XORW	036	Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants.
DOUBLE EXCLUSIVE OR	XORL	@XORL	612	Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants.

	XORW	XORL
Symbol	 XORW(036) I1 I2 R I1: Input 1 I2: Input 2 R: Result word	 XORL(612) I1 I2 R I1: Input 1 I2: Input 2 R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		XORW	XORL	XORW	XORL
I_1	Input 1	WORD	DWORD	1	2
I_2	Input 2	WORD	DWORD	1	2
R	Result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con- stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
XORW	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
XORL	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

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.

Function

● XORW

XORW(036) takes the logical exclusive OR of data specified in I_1 and I_2 and outputs the result to R.

$I_1 \bullet \overline{I_2} + \overline{I_1} \bullet I_2 \rightarrow R$

I_1	I_2	R
1	1	0
1	0	1
0	1	1
0	0	0

● XORL

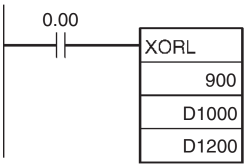
XORL(612) takes the logical exclusive OR of data specified in I_1 and I_2 as double-word data and outputs the result to R, R+1.

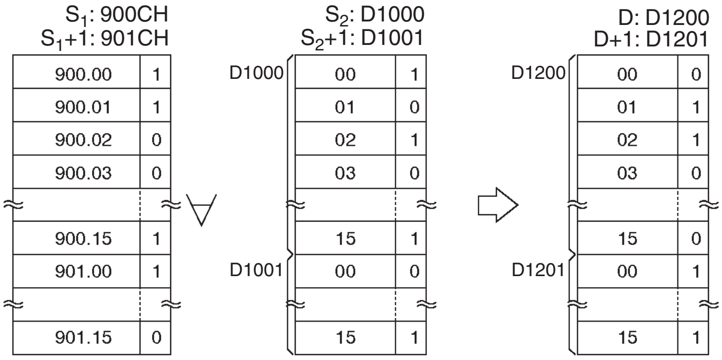
$(I_1, I_1+1) \bullet \overline{(I_2, I_2+1)} + \overline{(I_1, I_1+1)} \bullet (I_2, I_2+1) \rightarrow (R, R+1)$

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

Example Programming

When the execution condition CIO 0.00 is ON, the logical exclusive OR is taken of corresponding bits in CIO 901, CIO 900 and D1001, D1000 and the results will be output to corresponding bits in D1201 and D1200.





S_1: 900CH
S_1+1: 901CH

900.00	1
900.01	1
900.02	0
900.03	0
≈	≈
900.15	1
901.00	1
≈	≈
901.15	0

S_2: D1000
S_2+1: D1001

D1000	00	1
	01	0
	02	1
	03	0
	≈	≈
	15	1
D1001	00	0
	≈	≈
	15	1

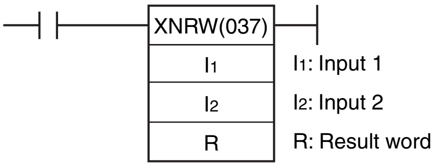
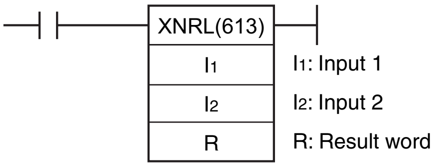
D: D1200
D+1: D1201

D1200	00	0
	01	1
	02	1
	03	0
	≈	≈
	15	0
D1201	00	1
	≈	≈
	15	1

Note: The symbol indicates exclusive logical OR.

XNRW/XNRL

Instruction	Mnemonic	Variations	Function code	Function
EXCLUSIVE NOR	XNRW	@XNRW	037	Takes the logical exclusive NOR of corresponding single words of word data and/or constants.
DOUBLE EXCLUSIVE NOR	XNRL	@XNRL	613	Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants.

	XNRW	XNRL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		XNRW	XNRL	XNRW	XNRL
I_1	Input 1	WORD	DWORD	1	2
I_2	Input 2	WORD	DWORD	1	2
R	Result word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
XNRW	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
	R											---							
XNRL	I_1, I_2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
	R											---							

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.

Function

● XNRW

XNRW(037) takes the logical exclusive NOR of data specified in I_1 and I_2 and outputs the result to R.

$$I_1 \bullet I_2 + \overline{I_1} \bullet \overline{I_2} \rightarrow R$$

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	1

● XNRL

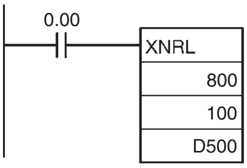
XNRL(613) takes the logical exclusive NOR of data specified in I_1 and I_2 and outputs the result to R, R+1.

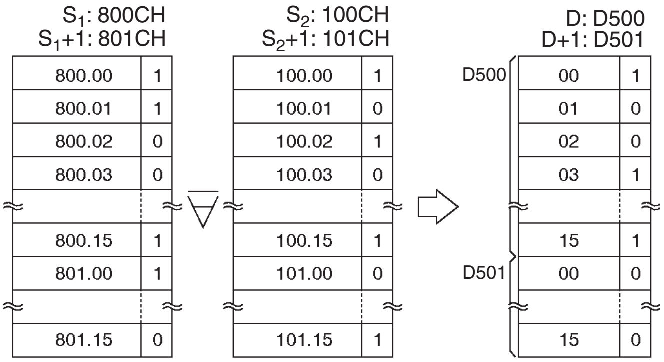
$$(I_1, I_1+1) \bullet (I_2, I_2+1) + \overline{(I_1, I_1+1)} \bullet \overline{(I_2, I_2+1)} \rightarrow (R, R+1)$$

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	0
0	1	0
0	0	1

Example Programming

When the execution condition CIO 0.00 is ON, the logical exclusive NOR is taken of corresponding bits in CIO 801, CIO 800, and CIO 101, CIO 100 and the results will be output to corresponding bits in D501 and D500.

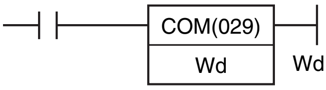
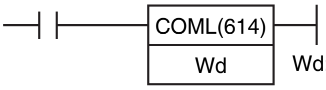




Note: The symbol indicates exclusive logical NOR.

COM/COML

Instruction	Mnemonic	Variations	Function code	Function
COMPLEMENT	COM	@COM	029	Turns OFF all ON bits and turns ON all OFF bits in Wd.
DOUBLE COMPLEMENT	COML	@COML	614	Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.

	COM	COML
Symbol	 Wd: Word	 Wd: Word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		COM	COML	COM	COML
Wd	Word	WORD	DWORD	1	2

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
COM	Wd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
COML	Wd												---						

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.

Function

● COM

COM(029) reverses the status of every specified bit in Wd.

$\overline{\text{Wd}} \rightarrow \text{Wd}$: 1 → 0 and 0 → 1

Note When using the COM instruction, be aware that the status of each bit will change each cycle in which the execution condition is ON.

● COML

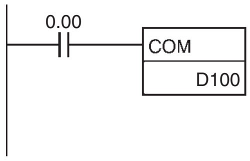
COML(614) reverses the status of every specified bit in Wd and Wd+1.

$(\overline{\text{Wd+1, Wd}}) \rightarrow (\text{Wd+1, Wd})$

Note When using the COML instruction, be aware that the status of each bit will change each cycle in which the execution condition is ON.

Example Programming

When CIO 0.00 is ON in the following example, the status of each bit D100 will be reversed.

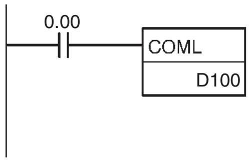


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D100	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

↓

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D100	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0

When CIO 0.00 is ON in the following example, the status of each bit in D100 and D101 will be reversed.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D100	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

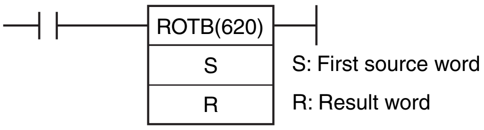
↓ ↓

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D100	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0
	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0

Special Math Instructions

ROTB

Instruction	Mnemonic	Variations	Function code	Function
BINARY ROOT	ROTB	@ROTB	620	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.

	ROTB
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	UDINT	2
R	Result word	UINT	1

● Operand Specifications

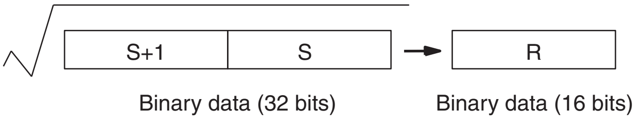
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---	OK						

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.
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

Function

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.



Note 1 The non-integer remainder is eliminated.

2 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.

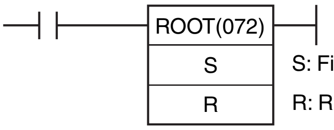
3 The operands of this instruction (S+1, S, and R) are all treated as binary values. If the input data is BCD, use the ROOT(072) instruction.

Example Programming

When CIO 0.00 is ON in the following example, ROTB(620) calculates the square root of the data in CIO 2 and CIO 1, and writes the integer portion of the result in D100.

ROOT

Instruction	Mnemonic	Variations	Function code	Function
BCD SQUARE ROOT	ROOT	@ROOT	072	Computes the square root of an 8-digit BCD number and outputs the integer portion of the result to the specified result word.

	ROOT
Symbol	 S: First source word R: Result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	DWORD	2
R	Result word	WORD	1

● Operand Specifications

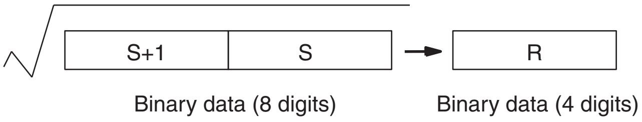
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---	OK						

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.

Function

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.

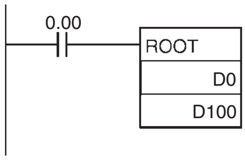


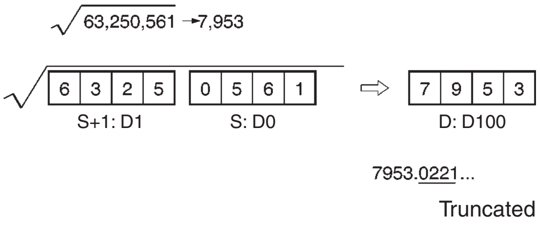
Note 1 The non-integer remainder is eliminated.

2 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.

Example Programming

When CIO 0.00 is ON in the following example, ROOT(072) calculates the square root of the data in D1 and D0, and writes the integer portion of the result in D100.



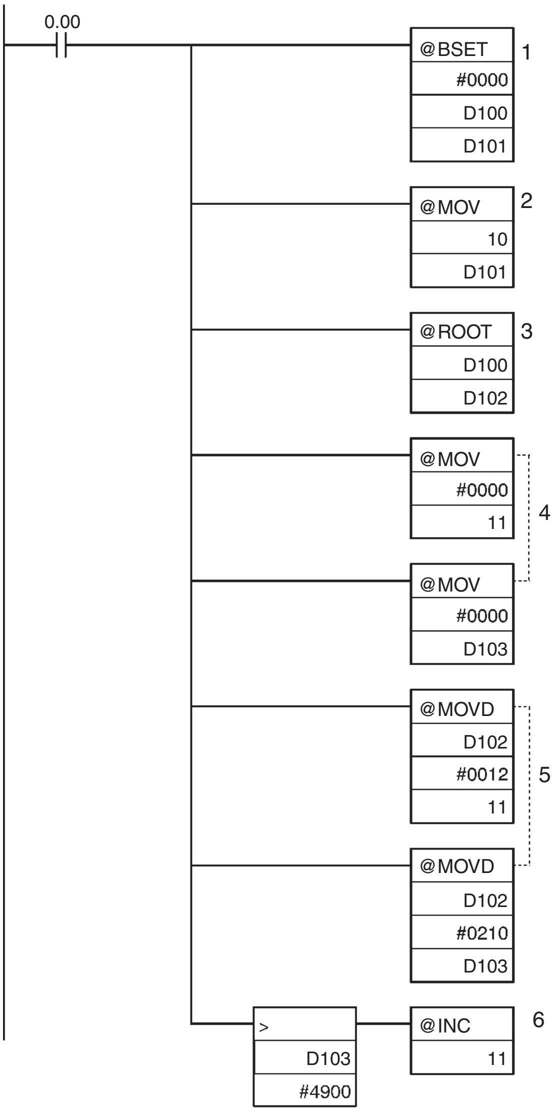


Application example

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 10, rounds off the result, and writes it to CIO 11.

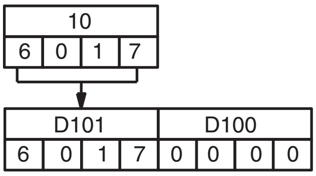
$\sqrt{6017} = 77.\underline{56}\ldots \rightarrow 78$

The values after the decimal point should be rounded.

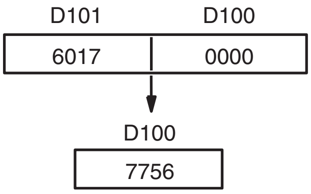


A ROOT instruction applies to 8-digit data, and thus data memory D101 and D100 are used.

1. The source words (D101 and D100) are cleared to 0000 0000.
2. The 4-digit number is moved to D101.



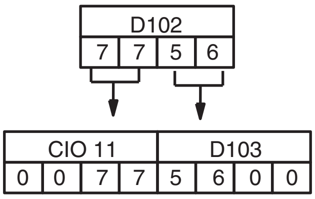
3. ROOT(072) calculates the square root of D101 and D100 and writes the result to D102.



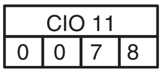
$\sqrt{60{,}170{,}000} = 7{,}756.932\ldots$

Square root computation
(Remainder eliminated)

4. D103 and the result word, CIO 11, are cleared to 0000.
5. The result of the square root calculation is divided by 100, with the integer portion written to CIO 11 and the remainder going to D103.



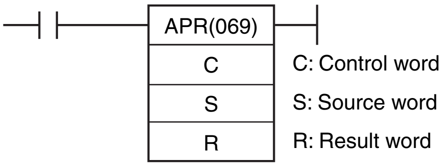
6. If the content of D103 is greater than 4900, CIO 11 is incremented by 1.



In this case, the result is 78.

APR

Instruction	Mnemonic	Variations	Function code	Function
ARITHMETIC PROCESS	APR	@APR	069	Calculates the sine, cosine, or a linear extrapolation of the source data.

	APR
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	UINT	Variable
S	Source data	WORD	1
R	Result word	WORD	1

● Sine Function

Operand	Value	Data range
C	0000 hex	---
S	0000 to 0900 (BCD)	0° to 90°
R	0000 to 0900 (BCD) 9999 (BCD)	0.0000 to 0.9999 1.0000

- 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

Operand	Value	Data range
C	0001 hex	---
S	0000 to 0900 (BCD)	0° to 90°
R	0000 to 0900 (BCD) 9999 (BCD)	0.0000 to 0.9999 1.0000

- 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.

Note The actual result for SIN(90°) and COS(0°) is 1, but 9999 (0.9999) will be output to R.

● Linear Extrapolation Function

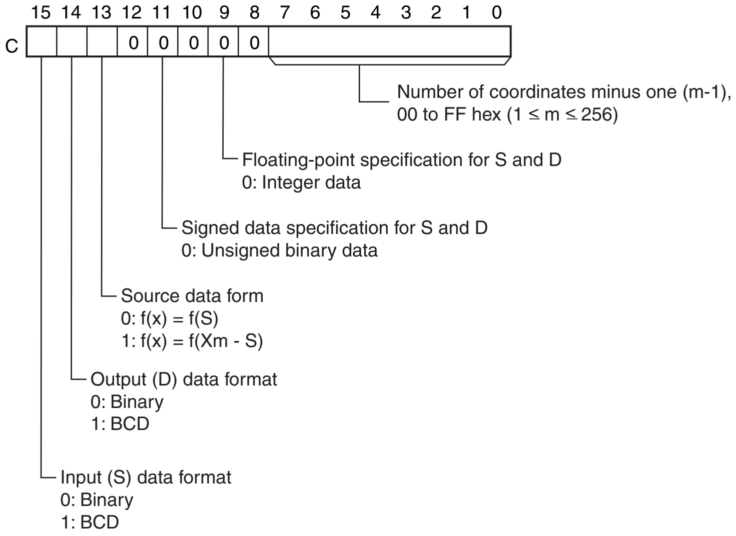
Operand	Value	Data range
C	Data area address	---
S	16-bit unsigned BCD data	0000 to 9999
	16-bit unsigned binary data	0 to 65,535
	16-bit signed binary data	-32,768 to 32,767
	32-bit signed binary data	-2,147,483,648 to 2,147,483,647
	Floating-point data	$-\infty$, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, $+\infty$
R	16-bit unsigned BCD data	0000 to 9999
	16-bit unsigned binary data	0 to 65,535
	16-bit signed binary data	-32,768 to 32,767
	32-bit signed binary data	-2,147,483,648 to 2,147,483,647
	Floating-point data	$-\infty$, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, $+\infty$

- Linear Extrapolation (C = Data area address)
  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.

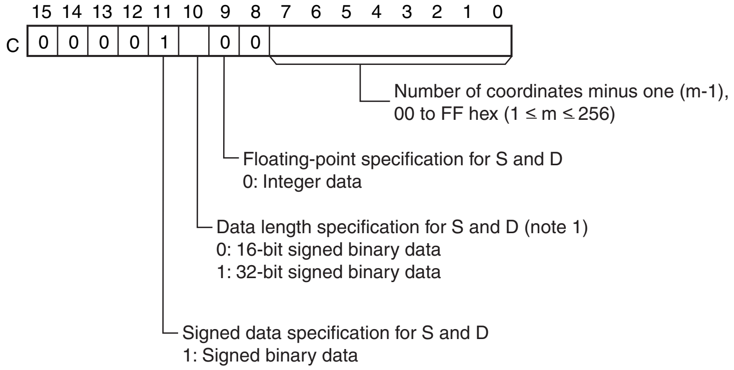
The following 5 kinds of I/O data can be used:

- 16-bit unsigned BCD data
- 16-bit unsigned binary data
- 16-bit signed binary data (CJ2/CS1-H/CJ1-H/CJ1M/CS1D Only)
- 32-bit signed binary data (CJ2/CS1-H/CJ1-H/CJ1M/CS1D Only)
- Single-precision floating-point data (CJ2/CS1-H/CJ1-H/CJ1M/CS1D Only)

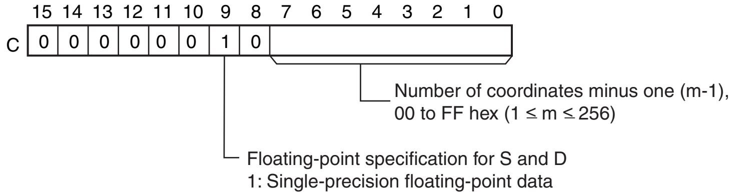
· Unsigned Integer Data (Binary or BCD)



· Signed Integer Data (Binary)



· Single-precision Floating-point Data



16-bit BCD16-bit binary (signed or unsigned) or 16-bit BCD data

C+1	X0 (*1)
C+2	Y0
C+3	X1
C+4	Y1
C+5	X2
C+6	Y2
	Xn
	Yn
C+ (2m+1)	Xm
C+ (2m+2)	Ym

Note: Write Xm (max. X value in the table) in word C+1 when the I/O data in S and D contain unsigned data (bit 11 of C = 0).

32-bit signed binary data

C+1	X0 (rightmost 16 bits)
C+2	X0 (leftmost 16 bits)
C+3	Y0 (rightmost 16 bits)
C+4	Y0 (leftmost 16 bits)
C+5	X1 (rightmost 16 bits)
C+6	X1 (leftmost 16 bits)
C+7	Y1 (rightmost 16 bits)
C+8	Y1 (leftmost 16 bits)
to	to
C+ (4n+1)	Xn (rightmost 16 bits)
C+ (4n+2)	Xn (leftmost 16 bits)
C+ (4n+3)	Yn (rightmost 16 bits)
C+ (4n+4)	Yn (leftmost 16 bits)
to	to
C+ (4m+1)	Xm (rightmost 16 bits)
C+ (4m+2)	Xm (leftmost 16 bits)
C+ (4m+3)	Ym (rightmost 16 bits)
C+ (4m+4)	Ym (leftmost 16 bits)

Floating-point data

C+1	X0 (rightmost 16 bits)
C+2	X0 (leftmost 16 bits)
C+3	Y0 (rightmost 16 bits)
C+4	Y0 (leftmost 16 bits)
C+5	X1 (rightmost 16 bits)
C+6	X1 (leftmost 16 bits)
C+7	Y1 (rightmost 16 bits)
C+8	Y1 (leftmost 16 bits)
to	to
C+ (4n+1)	Xn (rightmost 16 bits)
C+ (4n+2)	Xn (leftmost 16 bits)
C+ (4n+3)	Yn (rightmost 16 bits)
C+ (4n+4)	Yn (leftmost 16 bits)
to	to
C+ (4m+1)	Xm (rightmost 16 bits)
C+ (4m+2)	Xm (leftmost 16 bits)
C+ (4m+3)	Ym (rightmost 16 bits)
C+ (4m+4)	Ym (leftmost 16 bits)

Note: The X coordinates must be in ascending order: $X_1 < X_2 < ... < X_m$. Input all values of (X_n, Y_n) as binary data, regardless of the data format specified in control word C.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
S												OK						
R											---							

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 0. • OFF in all other cases.
Negative Flag	N	• ON if bit 15 of R is ON. • OFF in all other cases.

Function

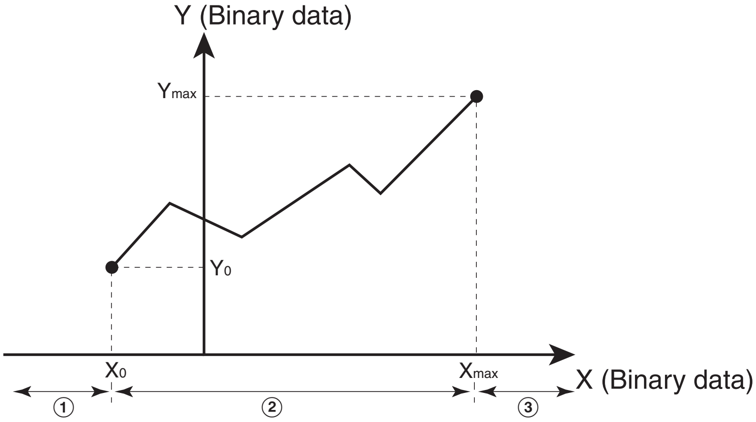
● Operation of the Linear Extrapolation Function

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.

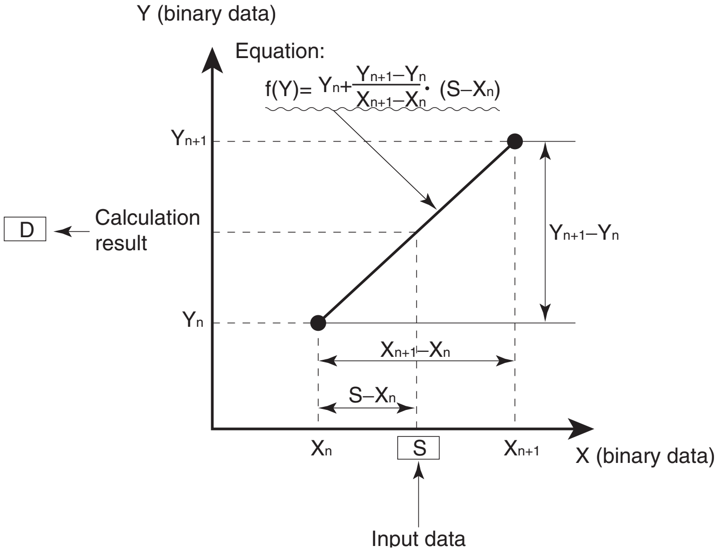
1. For $S < X_0$
Converted value = Y_0

2. For $X_0 \leq S \leq X_{max}$, if $X_n < S < X_{n+1}$
Converted value = $Y_n + [\{Y_n + 1 - Y_n\}/\{X_n + 1 - X_n\}] \times \{\text{Input data } S - X_n\}$

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.



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 (CJ2, CS1-H, CJ1-H, CJ1M, and CS1D Only)

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 (CJ2, CS1-H, CJ1-H, CJ1M, and CS1D Only)

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: 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 (CJ2, CS1-H, CJ1-H, CJ1M, and CS1D Only)

Setting name	Bit in C	Setting
Input data (S) format	15	0
Output data (D) format	14	0
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

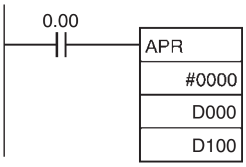
Note If the “Floating-point specification” in bit 09 of C is set to 1, a constant cannot be input for S.

Example Programming

Sine Function (C: #0000)

The following example shows APR(069) used to calculate the sine of 30°.

(SIN(30) = 0.5000)



Source data

S: D0			
0	$\times10^{1}$	$\times10^{0}$	$\times10^{-1}$
0	3	0	0

Set the source data in $\times10^{-1}$ degrees.
(0000 to 0900, BCD)

Result

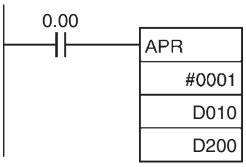
R: D100			
$\times10^{-1}$	$\times10^{-2}$	$\times10^{-3}$	$\times10^{-4}$
5	0	0	0

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°.

(COS(30) = 0.8660)



Source data

S: D10			
0	$\times10^{1}$	$\times10^{0}$	$\times10^{-1}$
0	3	0	0

Set the source data in $\times10^{-1}$ degrees.
(0000 to 0900, BCD)

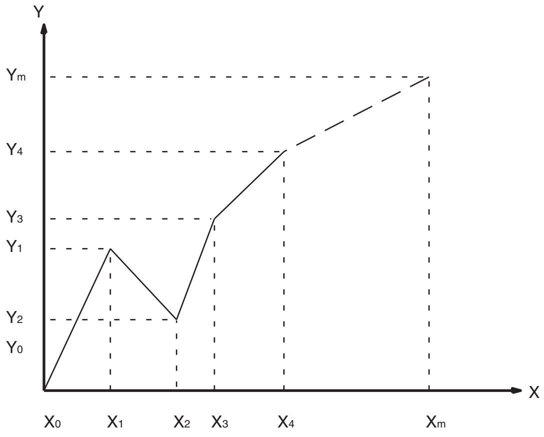
Result

R: D200			
$\times10^{-1}$	$\times10^{-2}$	$\times10^{-3}$	$\times10^{-4}$
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)

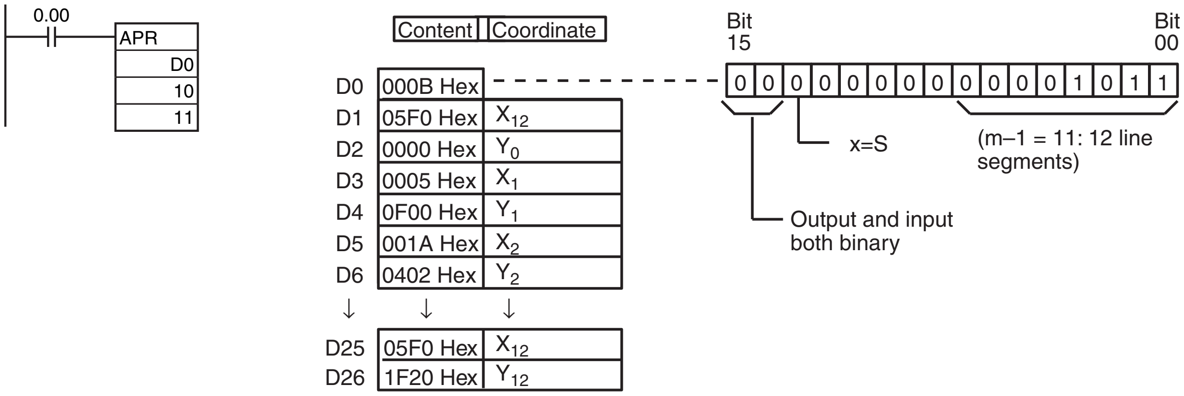
- Using 16-bit Unsigned BCD or Binary Data
  APR(069) processes the input data specified in S based on the control data in C and the line-segment data specified in the table beginning at C+1. The result is output to D.



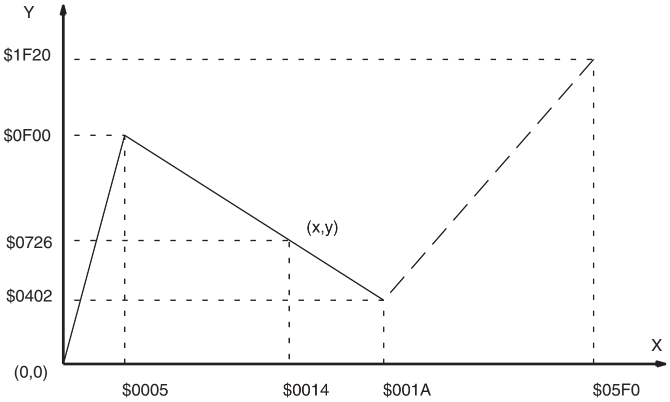
- $Y_n = f(X_n)$, $Y_0 = f(X_0)$
- Be sure that $X_{n-1} < X_n$ in all cases.
- Input all values of (X_n, Y_n) as binary data.

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

This example shows how to construct a linear extrapolation with 12 coordinates. The block of data is continuous, as it must be, from D0 to D26 (C to C + (2 × 12 + 2)). The input data is taken from CIO 10, and the result is output to CIO 11.



In this case, the source word, CIO 0010, contains 0014, and f(0014) = 0726 is output to R, CIO 0011.

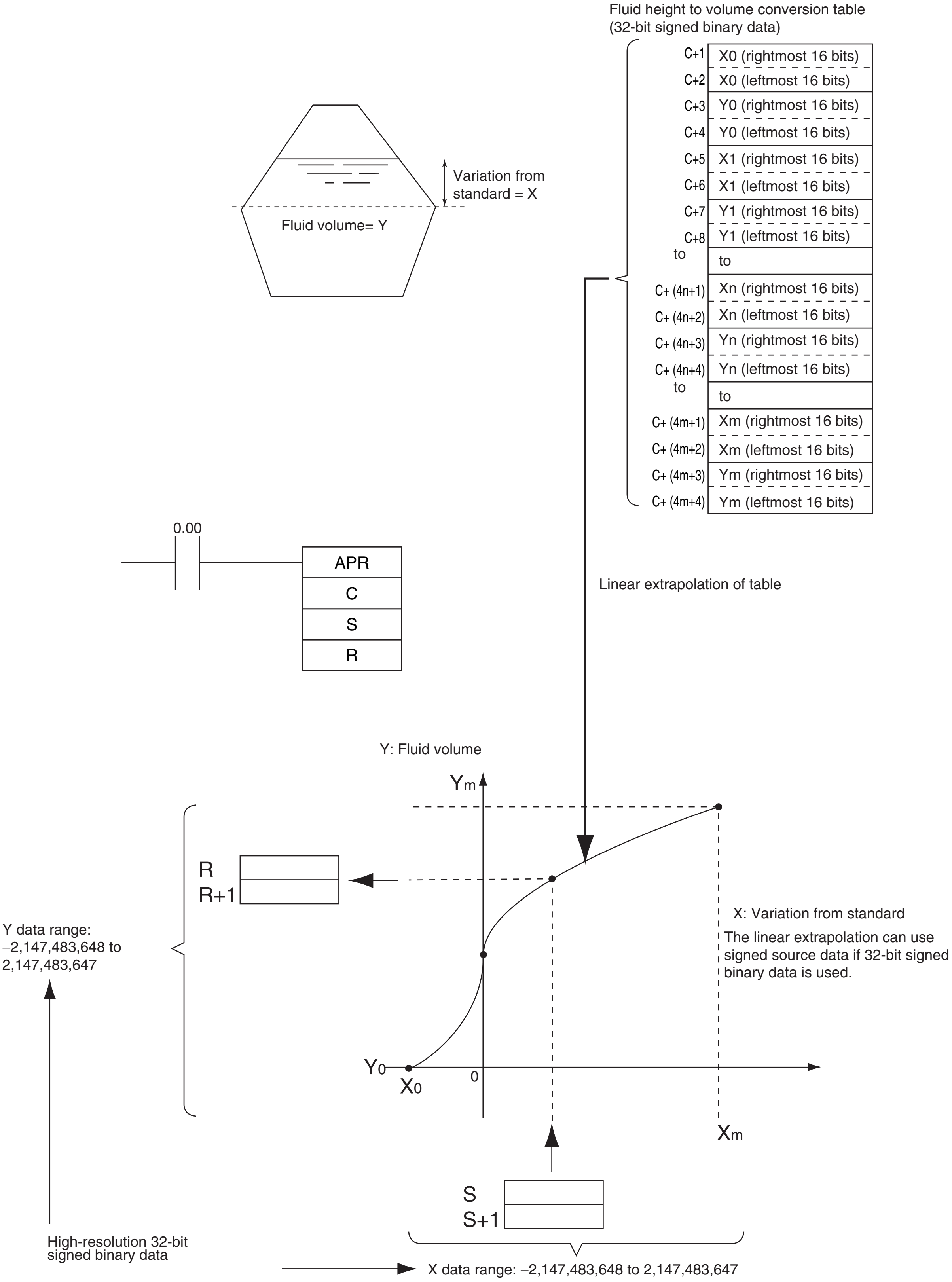


The linear-extrapolation calculation is shown below.

$$
\begin{aligned}
Y &= 0F00 + \frac{0402 - 0F00}{001A - 0005} \times (0014 - 0015) \\
&= 0F00 - (0086 \times 000F) \\
&= 0726
\end{aligned}
$$

Values are all hexadecimal (Hex).

- Using 32-bit Signed Binary Data (CJ2, CS1-H, CJ1-H, CJ1M, and CS1D ×Only)
  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.



Fluid height to volume conversion table (32-bit signed binary data)

C+1	X0 (rightmost 16 bits)
C+2	X0 (leftmost 16 bits)
C+3	Y0 (rightmost 16 bits)
C+4	Y0 (leftmost 16 bits)
C+5	X1 (rightmost 16 bits)
C+6	X1 (leftmost 16 bits)
C+7	Y1 (rightmost 16 bits)
C+8	Y1 (leftmost 16 bits)
to	to
C+ (4n+1)	Xn (rightmost 16 bits)
C+ (4n+2)	Xn (leftmost 16 bits)
C+ (4n+3)	Yn (rightmost 16 bits)
C+ (4n+4)	Yn (leftmost 16 bits)
to	to
C+ (4m+1)	Xm (rightmost 16 bits)
C+ (4m+2)	Xm (leftmost 16 bits)
C+ (4m+3)	Ym (rightmost 16 bits)
C+ (4m+4)	Ym (leftmost 16 bits)

Linear extrapolation of table

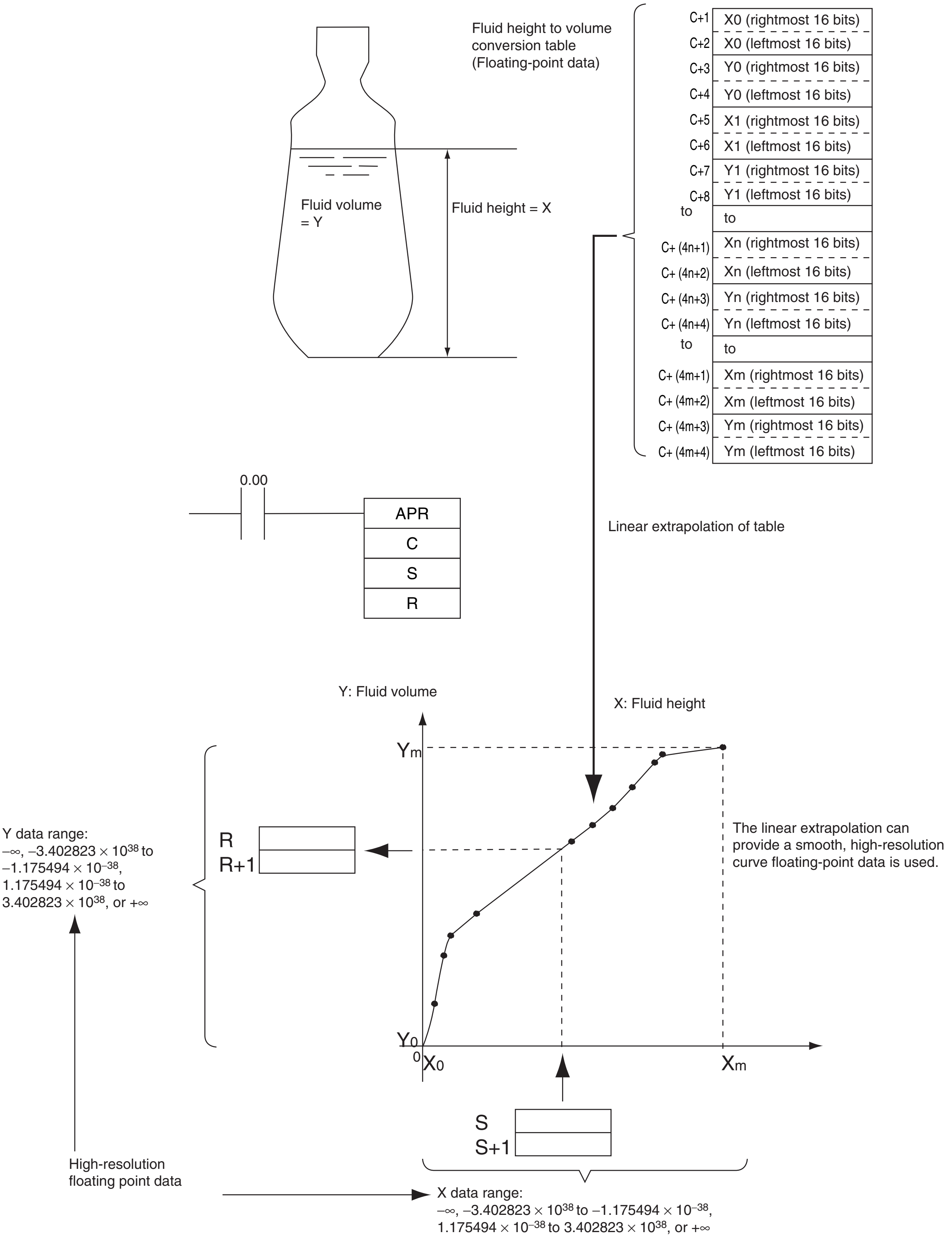
The linear extrapolation can use signed source data if 32-bit signed binary data is used.

Y data range: –2,147,483,648 to 2,147,483,647

X data range: –2,147,483,648 to 2,147,483,647

High-resolution 32-bit signed binary data

- Using Floating-point Data (CJ2, CS1-H, CJ1-H, CJ1M, and CS1D Only)
  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.



Fluid height to volume conversion table (Floating-point data)

C+1	X0 (rightmost 16 bits)
C+2	X0 (leftmost 16 bits)
C+3	Y0 (rightmost 16 bits)
C+4	Y0 (leftmost 16 bits)
C+5	X1 (rightmost 16 bits)
C+6	X1 (leftmost 16 bits)
C+7	Y1 (rightmost 16 bits)
C+8	Y1 (leftmost 16 bits)
to	to
C+ (4n+1)	Xn (rightmost 16 bits)
C+ (4n+2)	Xn (leftmost 16 bits)
C+ (4n+3)	Yn (rightmost 16 bits)
C+ (4n+4)	Yn (leftmost 16 bits)
to	to
C+ (4m+1)	Xm (rightmost 16 bits)
C+ (4m+2)	Xm (leftmost 16 bits)
C+ (4m+3)	Ym (rightmost 16 bits)
C+ (4m+4)	Ym (leftmost 16 bits)

Linear extrapolation of table

The linear extrapolation can provide a smooth, high-resolution curve floating-point data is used.

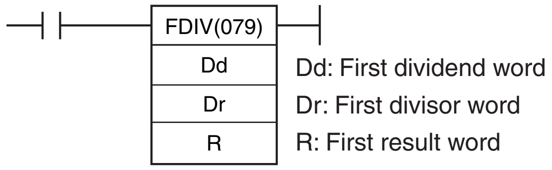
Y data range:
–∞, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, or +∞

High-resolution floating point data

X data range:
–∞, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, or +∞

FDIV

Instruction	Mnemonic	Variations	Function code	Function
FLOATING POINT DIVIDE (BCD)	FDIV	@FDIV	079	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).

	FDIV
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
Dd	First dividend word	UDINT	2
Dr	First divisor word	UDINT	2
R	First result word	UDINT	2

● Operand Specifications

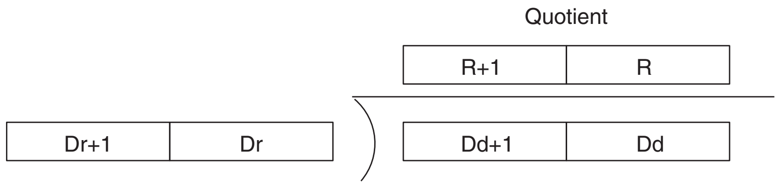
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
Dd	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
Dr																		
R																		

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.

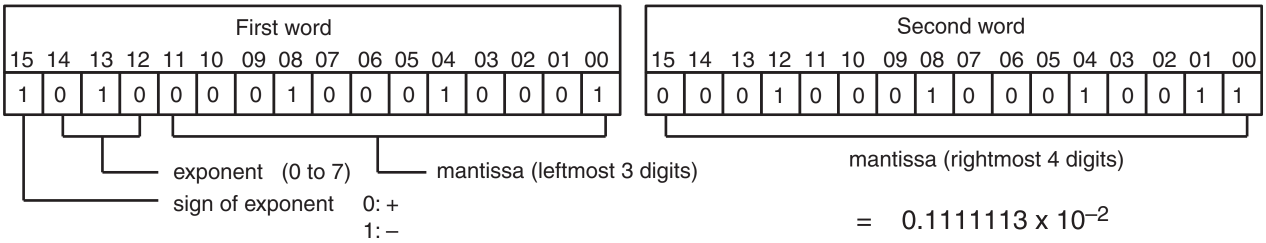
Function

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.



Note The result is expressed as a floating-point value, so it has 7 significant digits. The eighth and higher digits are eliminated.

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)

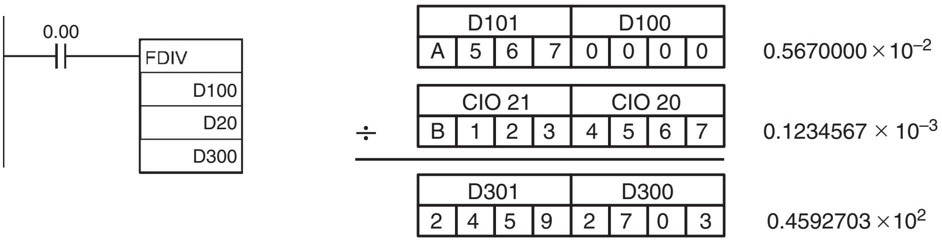
B123 4567: 0.1234567 × 10^{-3} (B = 1011 binary)

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}

Example Programming

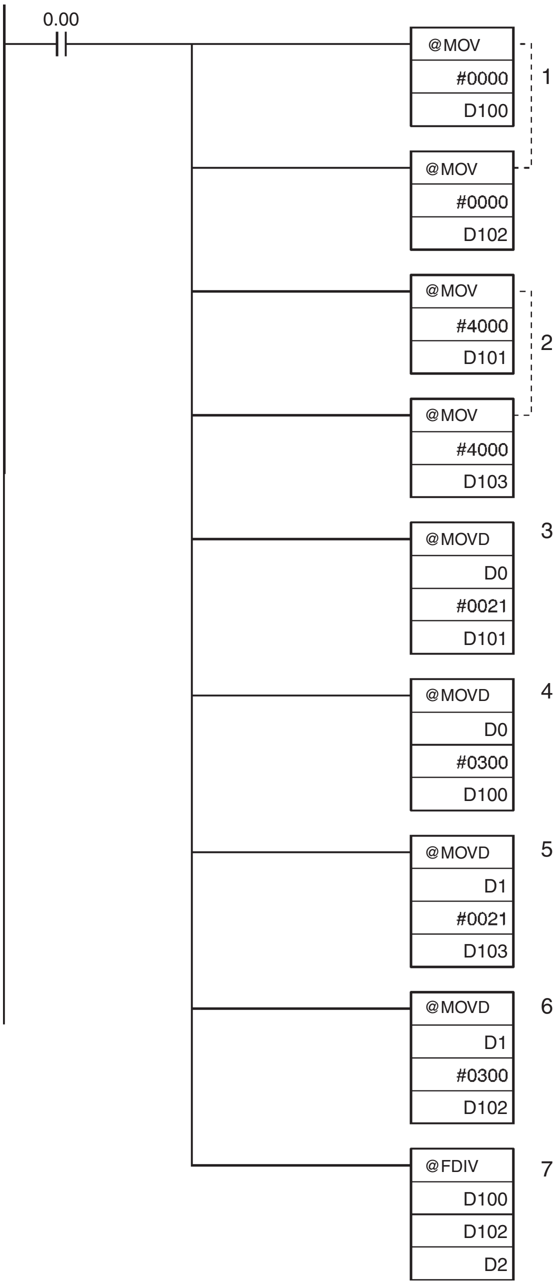
When CIO 0 is ON in the following example, FDIV(079) divides the floating-point number in D101 and D100 by the floating-point number in CIO 21 and CIO 20 and writes the result to D301 and D300.



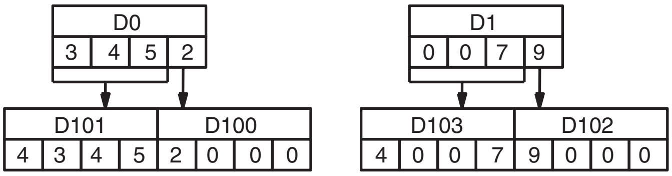
Application example

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 D3 and D2.

(Example) 0.3452×104÷0.0079×104=0.436962×102



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.



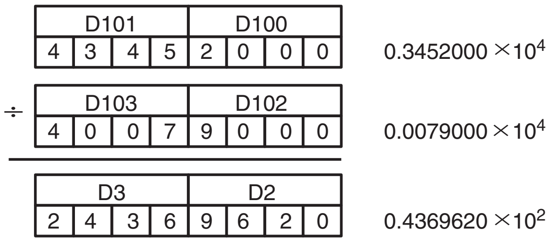
2. D101 and D103 are set to 4000.

1. D100 and D102 are set to 0000.

3 to 6. MOVD(083) is used to move the digits of the original source words to the proper digits in the 2-word floating-point formats.

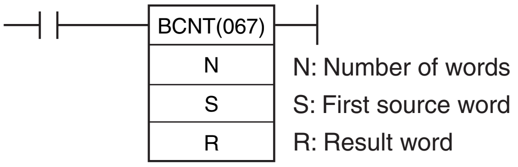
7. FDIV(079) divides the floating-point number in D101 and D100 by the floating-point number in D103 and D102.

The result of the calculation and the quotient are output to D3 and D2.



BCNT

Instruction	Mnemonic	Variations	Function code	Function
BIT COUNTER	BCNT	@BCNT	067	Counts the total number of ON bits in the specified word(s).

	BCNT
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Number of words	UINT	1
S	First source word	UINT	Variable
R	Result word	UINT	1

N: Number of words

The number of words must be 0001 to FFFF (1 to 65,535 words).

● Operand Specifications

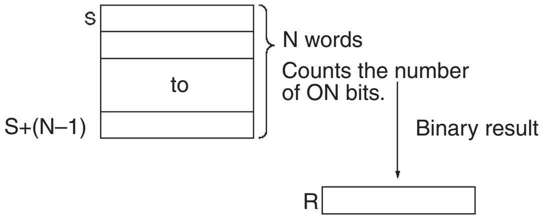
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
R												OK						

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.

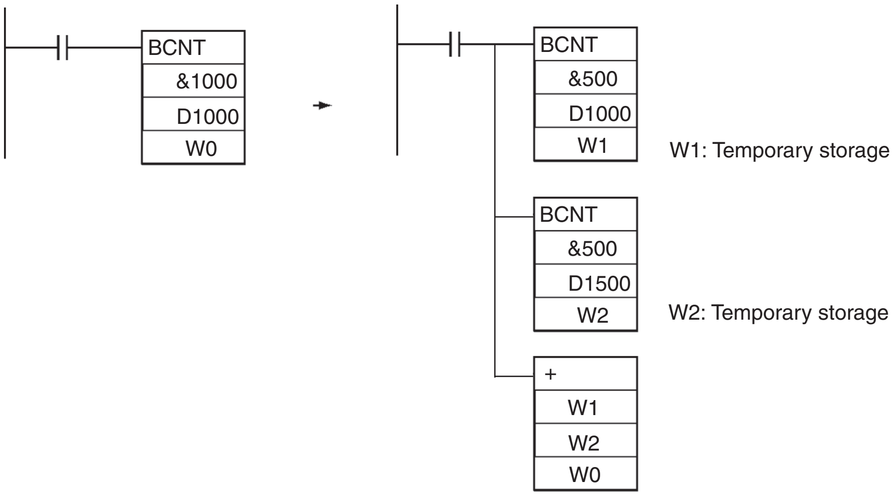
Function

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.



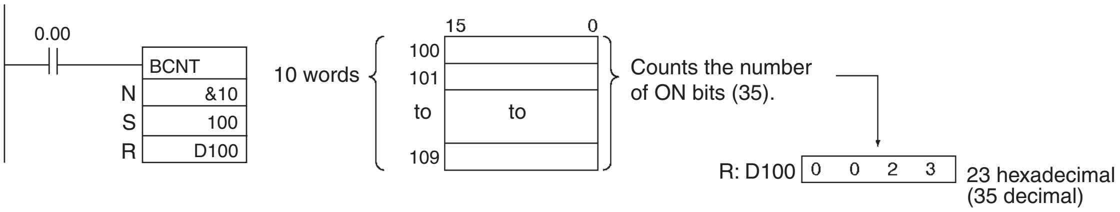
Precautions

- Some time will be required to complete BCNT(067) if a large number of words is specified. Even if an interrupt occurs, execution of this instruction will not be interrupted and execution of the interrupt task will be started after execution of BCNT(067) has been completed. One BCNT(067) instruction can be replaced with two BCNT(067) instructions to help avoid this problem.



Example Programming

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 100 through CIO 109 and writes the result to D100.



Floating-point Math Instructions

Floating-point Math Instructions

The Floating-point Math Instructions convert data and perform floating-point arithmetic operations.

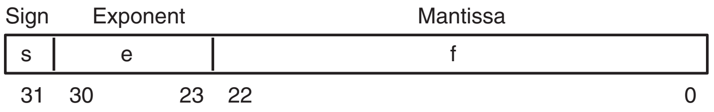
● Data Format

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.

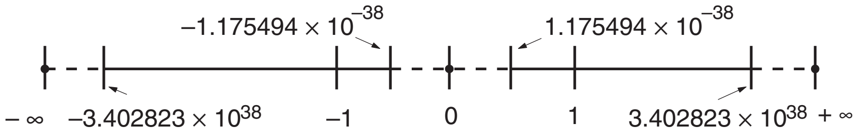
● Number of Digits

The number of effective digits for floating-point data is seven digits for decimal.

● Floating-point Data

The following data can be expressed by floating-point data:

- $-\infty$
- $-3.402823 \times 10^{38} \leq$ value $\leq -1.175494 \times 10^{-38}$
- 0
- $1.175494 \times 10^{-38} \leq$ 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

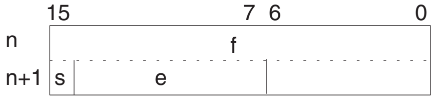
* NaN (not a number) is not a valid floating-point number. Executing floating-point calculation instructions will not result in NaN.

Writing Floating-point Constants

When writing floating-point constants, include the plus (+) or minus (–) sign.
For example, enter the following: +123, +123.45, or –123.45.
If you enter “&”, such as in “&123,” the value will be taken as a decimal value, and the floating-point instruction will not function correctly. If you enter “#”, such as in “#123,” the value will be treated in the following floating-point format: #00000123.

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.

(1) Normalized Numbers

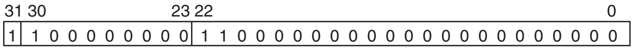
Normalized numbers express real numbers. The sign bit will be 0 for a positive number and 1 for a negative number.

The exponent (e) will be expressed from 1 to 254, and the real exponent will be 127 less, i.e., –126 to 127.

The mantissa (f) will be expressed from 0 to $2^{23} - 1$, and it is assume that, in the real mantissa, bit 2^{23} is 1 and the binary point follows immediately after it.

Normalized numbers are expressed as follows:
$(-1)^{(\text{sign s})} \times 2^{(\text{exponent e})-127} \times (1 + \text{mantissa} \times 2^{-23})$

Example



Sign: –
Exponent: 128 – 127 = 1
Mantissa: $1 + (2^{22} + 2^{21}) \times 2^{-23} = 1 + (2^{-1} + 2^{-2}) = 1 + 0.75 = 1.75$
Value: $-1.75 \times 2^{1} = -3.5$

(2) Non-normalized Numbers

Non-normalized numbers express real numbers with very small absolute values. The sign bit will be 0 for a positive number and 1 for a negative number.

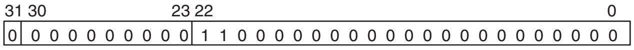
The exponent (e) will be 0, and the real exponent will be –126.

The mantissa (f) will be expressed from 1 to $2^{23} - 1$, and it is assume that, in the real mantissa, bit 2^{23} is 0 and the binary point follows immediately after it.

Non-normalized numbers are expressed as follows:

$(-1)^{(\text{sign s})} \times 2^{-126} \times (\text{mantissa x } 2^{-23})$

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}

(3) 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.

(4) Infinity

Values of $+\infty$ and $-\infty$ 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.

(5) NaN

NaN (not a number) is produced when the result of calculations, such as 0.0/0.0, ∞/∞, or $\infty-\infty$, 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

(1) 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.

(2) 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.

(3) 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.

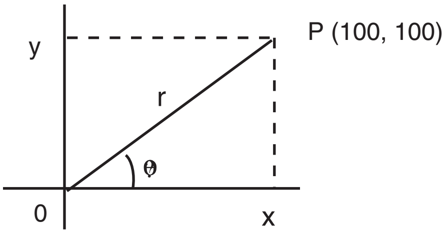
Floating-point Calculation Results

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 –∞.

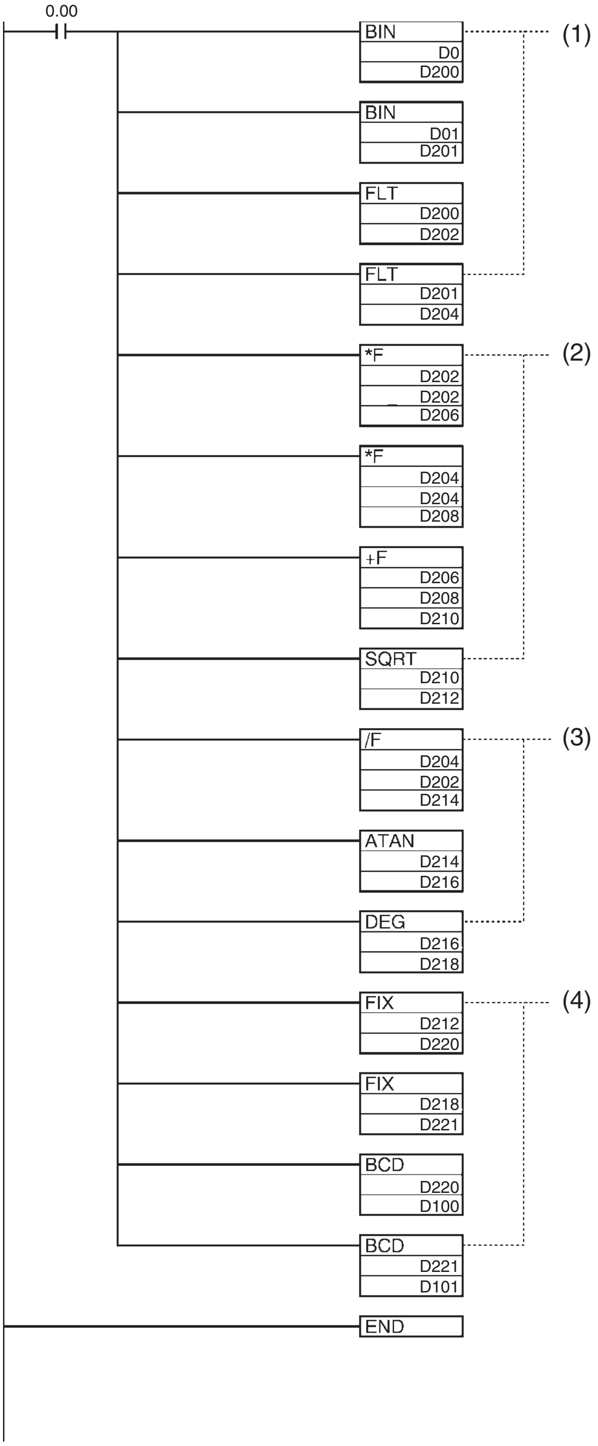
The Equals Flag will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow 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.



Example Programming



Calculations

$$\text{Distance } r = \sqrt{x^2 + y^2}$$

$$\text{Angle } \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

Example: Given the coordinates (100, 100), the distance r and the angle θ can be calculated from the above equation.

$$\text{Distance } r = \sqrt{100^2 + 100^2} = 141.4214$$

$$\text{Angle } \theta = \tan^{-1}\left(\frac{100}{100}\right) \times 180 \div \pi = 45.0$$

DM Contents

D0	#0100 (BCD)	x	→	D100	#0141 (BCD)	r
D1	#0100 (BCD)	y		D101	#0045 (BCD)	

(1) This section of the program converts the data from BCD to floating-point.

- The data area from D200 onwards is used as a work area.
- 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.
- The value of x that has been converted to floating-point data is output to D203 and D202.
- 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.

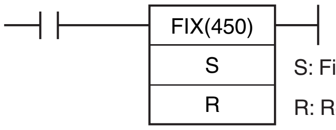
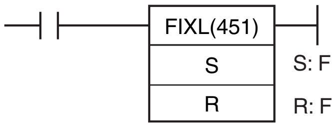
(3) In order to find the angle θ, Floating-point Math Instructions are used to calculate $\tan^{-1}$ (y/x). ATAN(465) outputs the result in radians, so DEG(459) is used to convert to degrees. The result is then output to D219 and D218 as floating-point data.

(4) The data is converted back from floating-point to BCD.

- 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.
- The distance r is output to D100.
- The angle θ is output to D101.

FIX/FIXL

Instruction	Mnemonic	Variations	Function code	Function
FLOATING TO 16-BIT	FIX	@FIX	450	Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word.
FLOATING TO 32-BIT	FIXL	@FIXL	451	Converts a 32-bit floating-point value to 32-bit signed binary data and places the result in the specified result words.

	FIX	FIXL
Symbol	 FIX(450) S R S: First source word R: Result word	 FIXL(451) S R S: First source word R: First result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		FIX	FIXL	FIX	FIXL
S	First source word	REAL	REAL	2	2
R	First result word	INT	DINT	1	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---	OK						

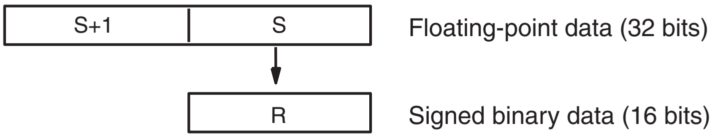
Flags

Name	Label	Operation
Error Flag	ER	• FIX ON if the integer portion of S+1 and S is not within the range of -32,768 to 32,767. • FIXL 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. • ON if the data in S+1 and S is not a number (NaN). • 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.

Function

● FIX

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.



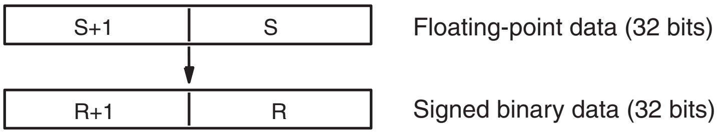
Only the integer portion of the floating-point data is converted, and the fraction portion is truncated.

Example conversions:

A floating-point value of 3.5 is converted to 3.
A floating-point value of –3.5 is converted to –3.

● FIXL

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.



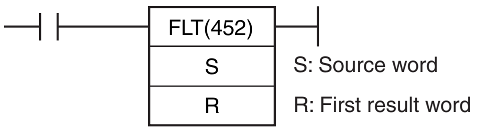
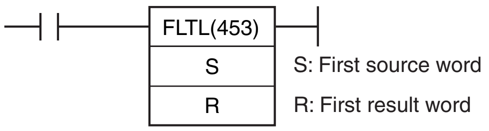
Only the integer portion of the floating-point data is converted, and the fraction portion is truncated.

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.

FLT/FLTL

Instruction	Mnemonic	Variations	Function code	Function
16-BIT TO FLOATING	FLT	@FLT	452	Converts a 16-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.
32-BIT TO FLOATING	FLTL	@FLTL	453	Converts a 32-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.

	FLT	FLTL
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		FLT	FLTL	FLT	FLTL
S	First source word	INT	DINT	1	2
R	First result word	REAL	REAL	2	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
R											---	---						

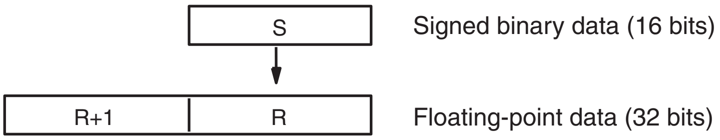
Flags

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.

Function

● FLT

FLT(452) converts the 16-bit signed binary value in S to 32-bit floating-point data (IEEE754-format) and places the result in R+1 and R. A single 0 is added after the decimal point in the floating-point result.



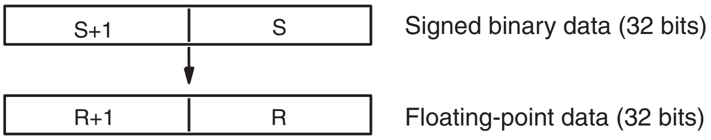
Only values within the range of –32,768 to 32,767 can be specified for S. To convert signed binary data outside of that range, use FLTL(453).

Example conversions:

A signed binary value of 3 is converted to 3.0.
A signed binary value of –3 is converted to –3.0.

● FLTL

FLTL(453) converts the 32-bit signed binary value in S+1 and S to 32-bit floating-point data (IEEE754-format) and places the result in R+1 and R. A single 0 is added after the decimal point in the floating-point result.



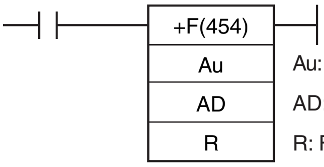
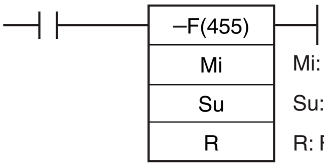
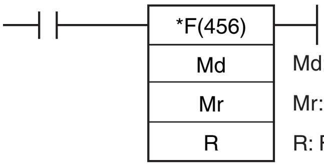
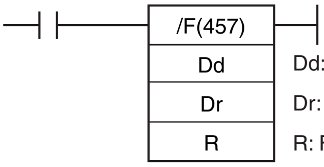
Signed binary data within the range of –2,147,483,648 to 2,147,483,647 can be specified for S+1 and S. The floating point value has 24 significant binary digits (bits). The result will not be exact if a number greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted by FLTL(453).

Example Conversions:

A signed binary value of 16,777,215 is converted to 16,777,215.0.
A signed binary value of –16,777,215 is converted to –16,777,215.0.

+F, –F, *F, /F

Instruction	Mnemonic	Variations	Function code	Function
FLOATING-POINT ADD	+F	@+F	454	Adds two 32-bit floating-point numbers and places the result in the specified result words.
FLOATING-POINT SUBTRACT	–F	@–F	455	Subtracts one 32-bit floating-point number from another and places the result in the specified result words.
FLOATING-POINT MULTIPLY	*F	@*F	456	Multiplies two 32-bit floating-point numbers and places the result in the specified result words.
FLOATING-POINT DIVIDE	/F	@/F	457	Divides one 32-bit floating-point number by another and places the result in the specified result words.

	+F	–F
Symbol	 Au: First augend word AD: First addend word R: First result word	 Mi: First Minuend word Su: First Subtrahend word R: First result word
	***F**	**/F**
	 Md: First Multiplicand word Mr: First Multiplier word R: First result word	 Dd: First Dividend word Dr: First Divisor word R: First result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand		Description	Data type	Size
+F	Au	First augend word	REAL	2
	AD	First addend word		
-F	Mi	First Minuend word	REAL	2
	Su	First Subtrahend word		
*F	Md	First Multiplicand word	REAL	2
	Mr	First Multiplier word		
/F	Dd	First Dividend word	REAL	2
	Dr	First Divisor word		
R		First result word	REAL	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
Au, AD, Mi, Su, Md, Mr, Dd, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

Refer to *Writing Floating-point Constants* on page 384 for information on floating-point expressions.

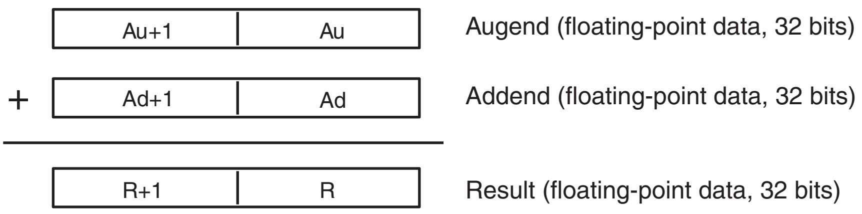
Flags

Name	Label	Operation
Error Flag	ER	ON if Au, AD, Mi, Su, Md, Dd is not recognized as floating-point data. **+F** • ON if the augend or addend data is not a number (NaN). • ON if +∞ and –∞ are added. **-F** • ON if the minuend or subtrahend is not a number (NaN). • ON if +∞ is subtracted from +∞. • ON if –∞ is subtracted from –∞. ***F** • ON if the multiplicand or multiplier is not a number (NaN). • ON if +∞ and 0 are multiplied. • ON if –∞ and 0 are multiplied. **/F** • 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. • OFF in all other cases.
Underflow Flag	UF	• ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value. • OFF in all other cases.
Negative Flag	N	• ON if the result is negative. • OFF in all other cases.

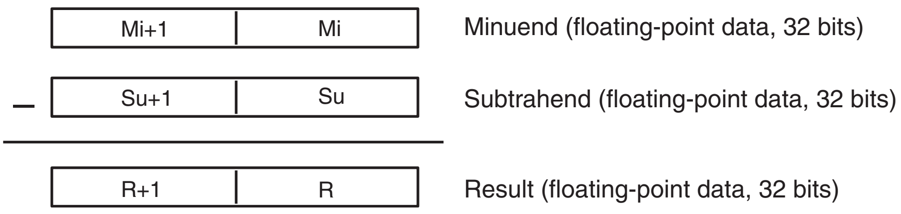
Function

The data specified in Au/Mi/Md/Dd and the data specified in AD/Su/Mr/Dr are added (+F), subtracted (-F), multiplied (*F), or divided (/F) as single-precision floating-point data (32 bits: IEEE754) and output to D+1, D.

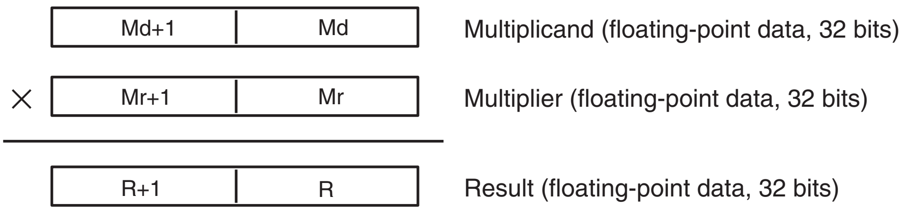
● +F



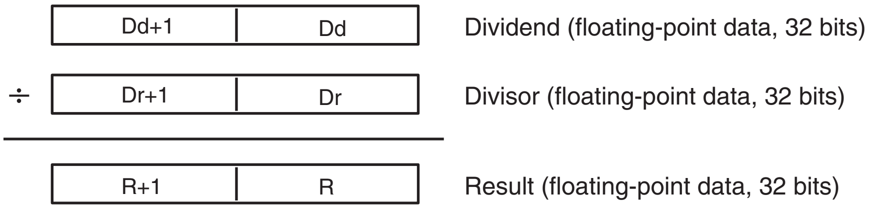
● -F



● *F



/F



- 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.

Operation rules

The result of an operation is output as shown below depending on the combination of floating-point data.

FLOATING-POINT ADD (+F)

	Augend				
Addend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞ (See note 2.)	–∞ (See note 2.)	
+∞	+∞	+∞ (See note 2.)	+∞	ER	
–∞	–∞	–∞ (See note 2.)	ER	–∞	
NaN					ER

Note 1 The results could be zero (including underflows), a numeral, +∞, or –∞.

2 With CJ1H-CPU□□H-R CPU Units, an undetermined value will be output.

ER The Error Flag will be turned ON and the instruction will not be executed.

FLOATING-POINT SUBTRACT (–F)

	Minuend				
Subtrahend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞ (See note 2.)	–∞ (See note 2.)	
+∞	–∞ (See note 2.)	–∞ (See note 2.)	ER	–∞	
–∞	+∞	+∞	+∞	ER	
NaN					ER

Note 1 The results could be zero (including underflows), a numeral, +∞, or –∞.

2 With CJ1H-CPU□□H-R CPU Units, an undetermined value will be output.

ER The Error Flag will be turned ON and the instruction will not be executed.

● FLOATING-POINT MULTIPLY (*F)

	Multiplicand				
Multiplier	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	0	ER	ER	
Numeral	0	See note 1.	+/–∞ (See note 2.)	+/–∞ (See note 2.)	
+∞	ER	+/–∞ (See note 2.)	+∞	–∞	
–∞	ER	+/–∞ (See note 2.)	–∞	+∞	
NaN					ER

Note 1 The results could be zero (including underflows), a numeral, +∞, or –∞.

2 With CJ1H-CPU□□H-R CPU Units, an undetermined value will be output.

ER The Error Flag will be turned ON and the instruction will not be executed.

● FLOATING-POINT DIVIDE (/F)

	Dividend				
Divisor	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	ER	+/–∞ (See note 3.)	+∞ (See note 3.)	–∞ (See note 3.)	
Numeral	0	See note 2.	+/–∞	+/–∞	
+∞	0	0 (See note 1 and 3.)	ER	ER	
–∞	0	0 (See note 1 and 3.)	ER	ER	
NaN					ER

Note 1 The results will be zero for underflows.

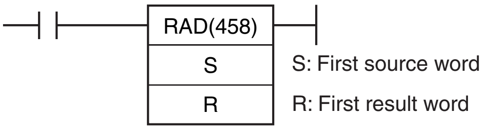
2 The results could be zero (including underflows), a numeral, +∞, or –∞.

3 With CJ1H-CPU□□H-R CPU Units, an undetermined value will be output.

ER The Error Flag will be turned ON and the instruction will not be executed.

RAD

Instruction	Mnemonic	Variations	Function code	Function
DEGREES TO RADIANS	RAD	@RAD	458	Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words.

Symbol	RAD
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

● Operand Specifications

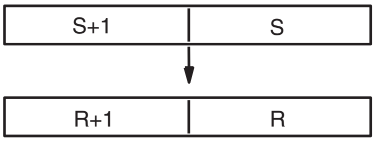
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

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. • OFF in all other cases.
Underflow Flag	UF	• ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value. • OFF in all other cases.
Negative Flag	N	• ON if the result is negative. • OFF in all other cases.

Function

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)

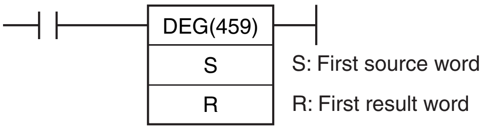
Degrees are converted to radians by means of the following formula:

Degrees × π/180 = radians

Note
- 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.

DEG

Instruction	Mnemonic	Variations	Function code	Function
RADIANS TO DEGREES	DEG	@DEG	459	Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words.

	DEG
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

● Operand Specifications

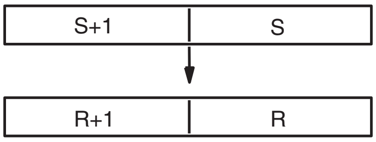
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

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. • OFF in all other cases.
Underflow Flag	UF	• ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value. • OFF in all other cases.
Negative Flag	N	• ON if the result is negative. • OFF in all other cases.

Function

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:

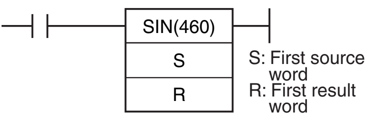
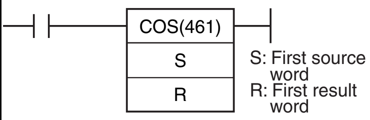
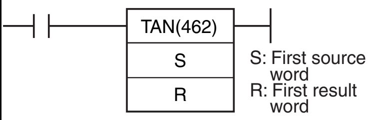
Radians × 180/π = degrees

Note

- 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.

SIN/COS/TAN

Instruction	Mnemonic	Variations	Function code	Function
SINE	SIN	@SIN	460	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.
COSINE	COS	@COS	461	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.
TANGENT	TAN	@TAN	462	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.

	SIN	COS	TAN
Symbol			

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

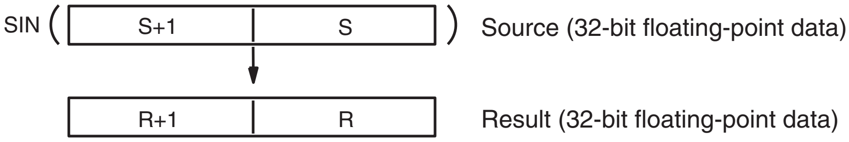
Flags

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 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.

Function

● SIN

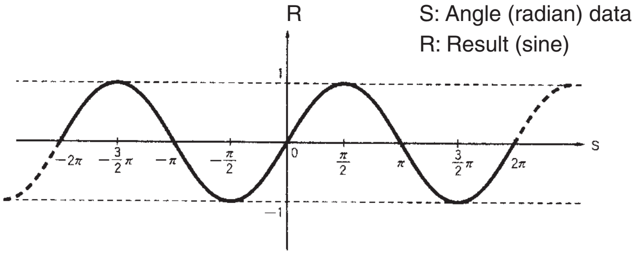
SIN(460) calculates the sine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S.

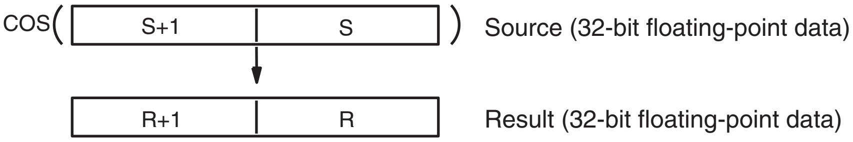
For information on converting from degrees to radians, see *3-15-22 LOGARITHM: LOG(468) DEGREES TO RADIANS: RAD(458)*.

The following diagram shows the relationship between the angle and result.



● COS

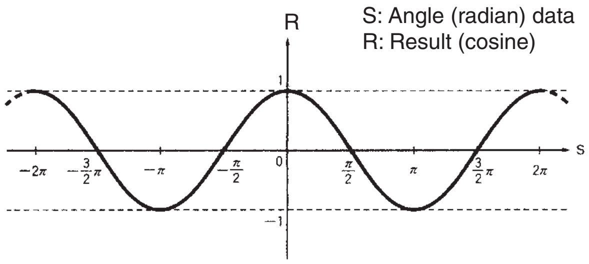
COS(461) calculates the cosine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (-65,535 to 65,535) in radians in S+1 and S.

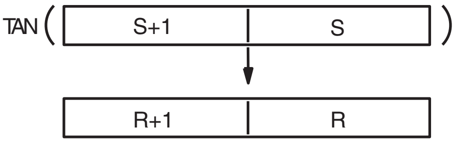
For information on converting from degrees to radians, see *3-15-22 LOGARITHM: LOG(468) DEGREES TO RADIANS: RAD(458)*.

The following diagram shows the relationship between the angle and result.



● TAN

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.)



Source (32-bit floating-point data)

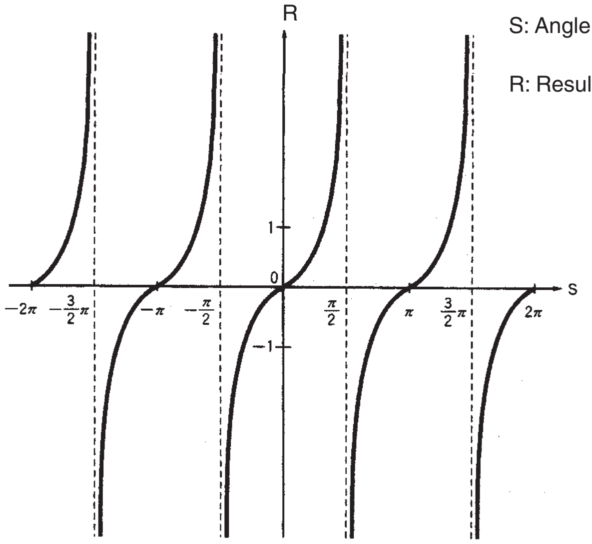
Result (32-bit floating-point data)

Specify the desired angle (-65,535 to 65,535) in radians in S+1 and S.

For information on converting from degrees to radians, see *3-15-22 LOGARITHM: LOG(468) 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.

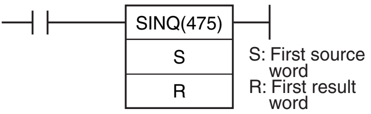
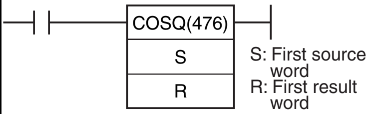
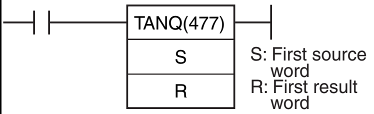


S: Angle (radian) data

R: Result (tangent)

SINQ/COSQ/TANQ

Instruction	Mnemonic	Variations	Function code	Function
HIGH-SPEED SINE	SINQ	@SINQ	475	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.
HIGH-SPEED COSINE	COSQ	@COSQ	476	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.
HIGH-SPEED TANGENT	TANQ	@TANQ	477	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.

	SINQ	COSQ	TANQ
Symbol			

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

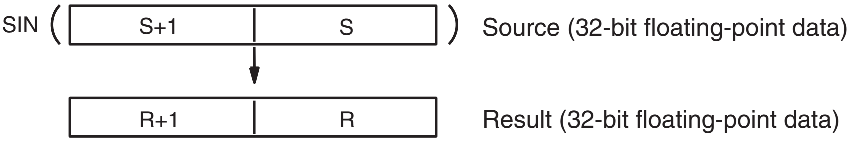
● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

Function

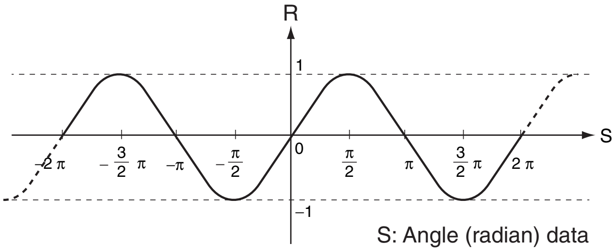
● SINQ

SINQ(475) 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.
  For information on converting between degrees and radians, see *3-15-9 DEGREES TO RADIANS: RAD(458)* and *3-15-10 RADIANS TO DEGREES: DEG(459)*.
- If the angle is outside of the range -65,535 to 65,535, an unpredictable value will be output, but the Error Flag will not be turned ON.

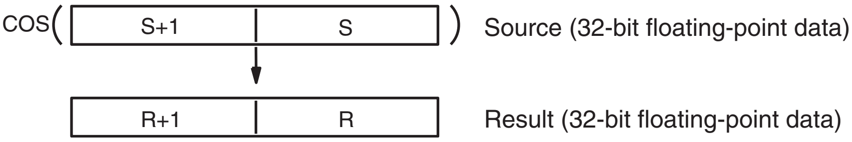
The following diagram shows the relationship between the input data and result.



S: Angle (radian) data
R: Result (sine)

● COSQ

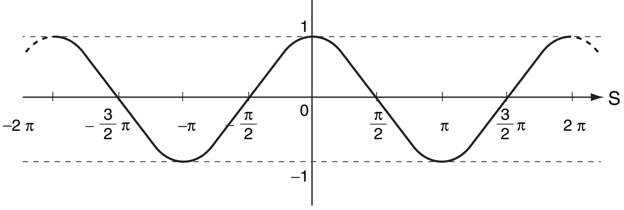
COSQ(476) calculates the cosine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R.
(The floating point source data must be in IEEE754 format.)



- Specify the desired angle (-65,535 to 65,535) in radians in S+1 and S.
  For information on converting between degrees and radians, see *3-15-9 DEGREES TO RADIANS: RAD(458)* and *3-15-10 RADIANS TO DEGREES: DEG(459)*.
- If the angle is outside of the range -65,535 to 65,535, an unpredictable value will be output, but the Error Flag will not be turned ON.

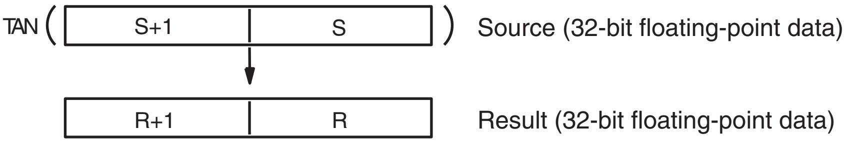
The following diagram shows the relationship between the input data and result.

S: Angle (radian) data
R: Result (cosine)



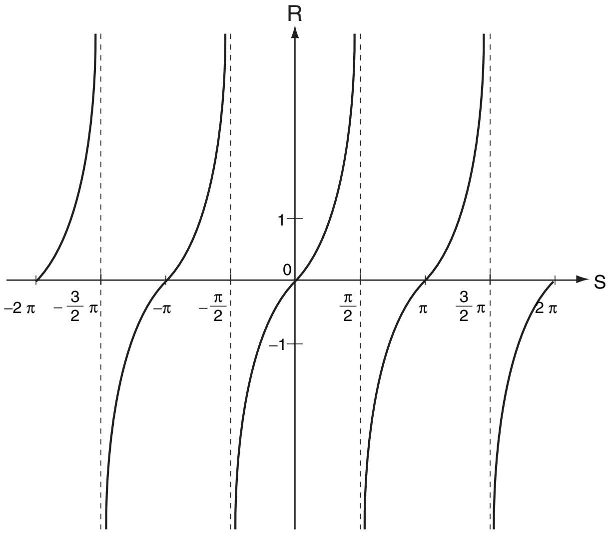
● TANQ

TANQ(477) 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.
  For information on converting between degrees and radians, see *3-15-9 DEGREES TO RADIANS: RAD(458)* and *3-15-10 RADIANS TO DEGREES: DEG(459)*.
- If the angle is outside of the range -65,535 to 65,535, an unpredictable value will be output, but the Error Flag will not be turned ON.
- If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the result will be output as ±∞ or 0.

The following diagram shows the relationship between the input data and result.



S: Angle (radian) data
R: Result (tangent)

Precautions

● SINQ, COSQ

SINQ(475)/COSQ(476) differs from SIN(460)/COS(461) in the following respects:

- The instruction has improved performance.
- The instruction length is 8 steps.
- The Condition Flags are not refreshed.
- An unpredictable value will be output if the angle data is out-of-range.
- The data cannot be input or output at a Programming Console. A question mark will be displayed.

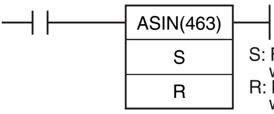
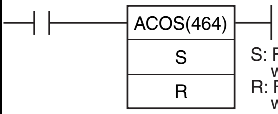
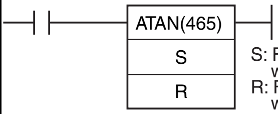
● TANQ

TANQ(477) differs from TAN(462) in the following respects:

- The instruction has improved performance.
- The instruction length is 15 steps.
- The Condition Flags are not refreshed.
- An unpredictable value will be output if the angle data is out-of-range.
- The data cannot be input or output at a Programming Console. A question mark will be displayed.
- An unpredictable value will be output if the angle data is $n\pi/2$ (n = ..., -3, -1, 1, 3.....).

ASIN/ACOS/ATAN

Instruction	Mnemonic	Variations	Function code	Function
ARC SINE	ASIN	@ASIN	463	Calculates the arc sine of a 32-bit floating-point number and places the result in the specified result words.
ARC COSINE	ACOS	@ACOS	464	Calculates the arc cosine of a 32-bit floating-point number and places the result in the specified result words.
ARC TANGENT	ATAN	@ATAN	465	Calculates the arc tangent of a 32-bit floating-point number and places the result in the specified result words.

	ASIN	ACOS	ATAN
Symbol	 S: First source word R: First result word	 S: First source word R: First result word	 S: First source word R: First result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	ASIN: SIN (sine) data rightmost word number ACOS: COS (cosine) data rightmost word number ATAN: TAN (tangent) data rightmost word number	REAL	2
R	First result word	REAL	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

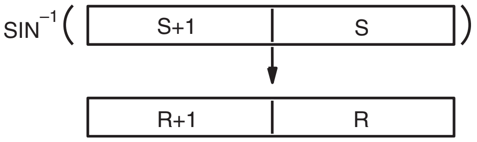
Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. • ASIN ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. • ACOS ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. • ATAN 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	• ASIN ON if the result is negative. OFF in all other cases. • ACOS OFF

Function

● ASIN

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.)

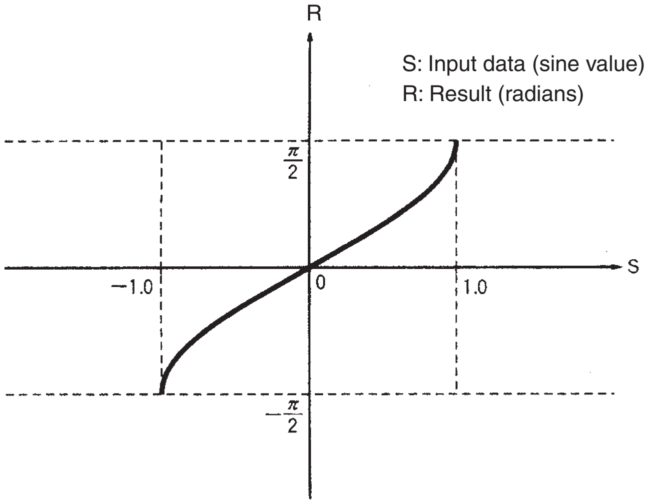


Source (32-bit floating-point data)

Result (32-bit floating-point data)

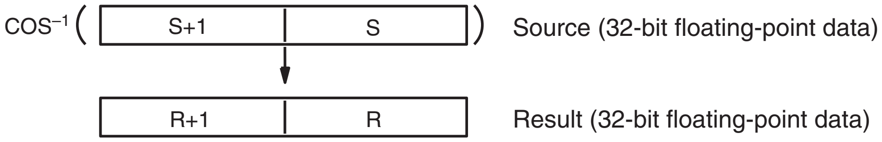
The result is output to words R+1 and R as an angle (in radians) within the range of –π/2 to π/2.

The following diagram shows the relationship between the input data and result.



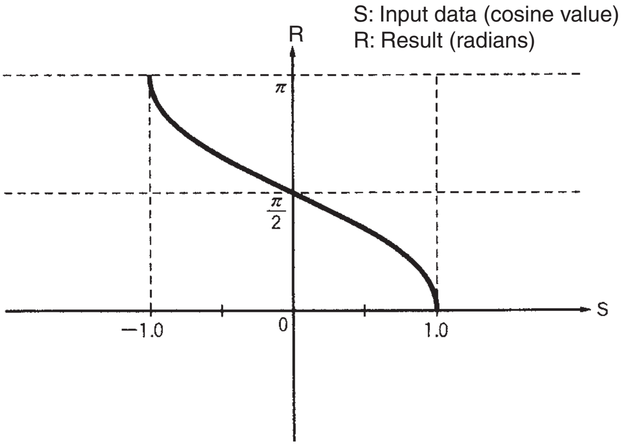
● ACOS

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.)



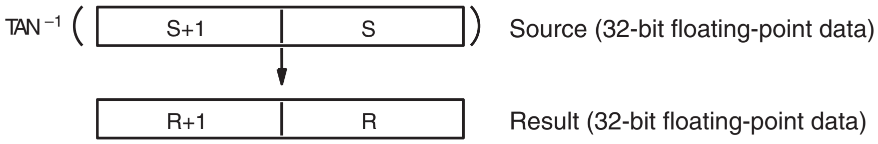
The result is output to words R+1 and R as an angle (in radians) within the range of 0 to π.

The following diagram shows the relationship between the input data and result.



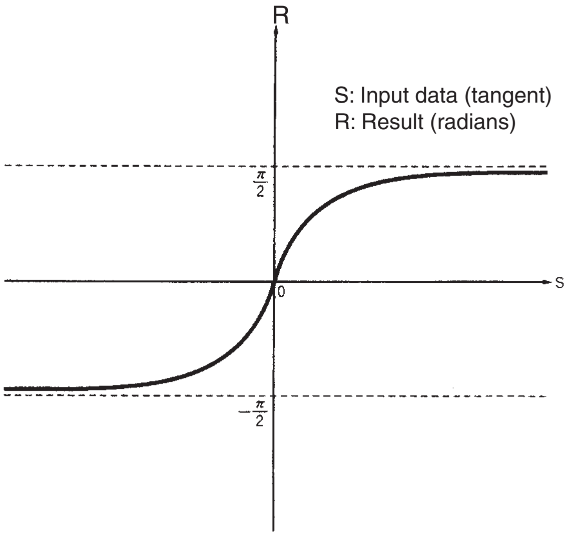
● ATAN

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.)



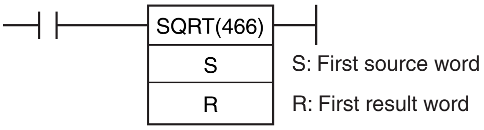
The result is output to words R+1 and R as an angle (in radians) within the range of –π/2 to π/2.

The following diagram shows the relationship between the input data and result.



SQRT

Instruction	Mnemonic	Variations	Function code	Function
SQUARE ROOT	SQRT	@SQRT	466	Calculates the square root of a 32-bit floating-point number and places the result in the specified result words.

Symbol	SQRT
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

● Operand Specifications

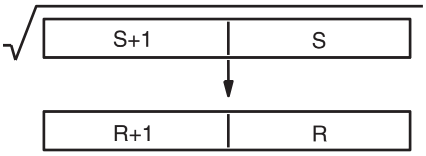
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

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. • OFF in all other cases.
Underflow Flag	UF	OFF
Negative Flag	N	OFF

Function

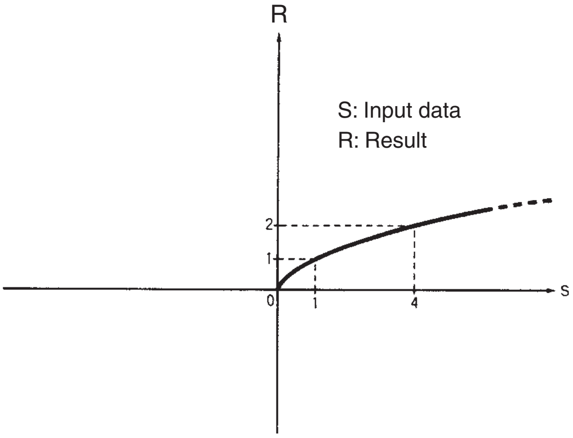
SQRT(466) calculates the square root of the 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Source (32-bit floating-point data)

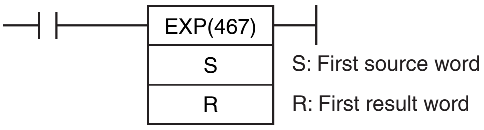
Result (32-bit floating-point data)

The following diagram shows the relationship between the input data and result.



EXP

Instruction	Mnemonic	Variations	Function code	Function
EXPONENT	EXP	@EXP	467	Calculates the natural (base e) exponential of a 32-bit floating-point number and places the result in the specified result words.

	EXP
Symbol	 S: First source word R: First result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

● Operand Specifications

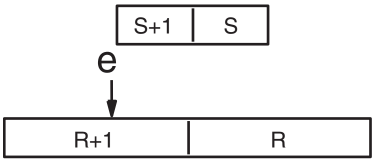
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

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. • OFF in all other cases.
Underflow Flag	UF	• ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value. • OFF in all other cases.
Negative Flag	N	OFF

Function

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.

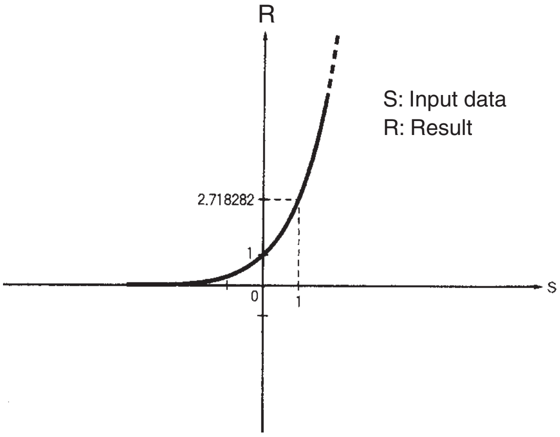


Source (32-bit floating-point data)

Result (32-bit floating-point data)

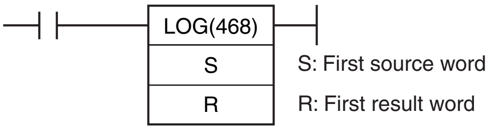
The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



LOG

Instruction	Mnemonic	Variations	Function code	Function
LOGARITHM	LOG	@LOG	468	Calculates the natural (base e) logarithm of a 32-bit floating-point number and places the result in the specified result words.

	LOG
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
R	First result word	REAL	2

● Operand Specifications

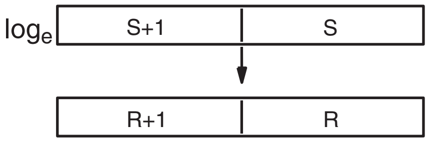
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

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. • OFF in all other cases.
Underflow Flag	UF	OFF
Negative Flag	N	• ON if the result is negative. • OFF in all other cases.

Function

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.

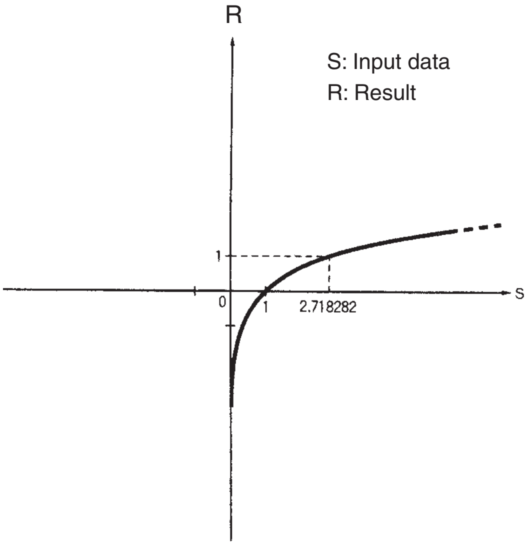


Source (32-bit floating-point data)

Result (32-bit floating-point data)

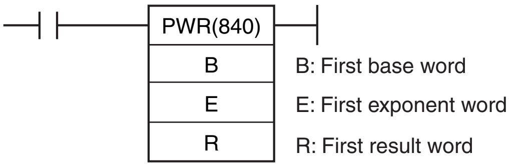
The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



PWR

Instruction	Mnemonic	Variations	Function code	Function
EXPONENTIAL POWER	PWR	@PWR	840	Raises a 32-bit floating-point number to the power of another 32-bit floating-point number.

	PWR
Symbol	

B: First base word
E: First exponent word
R: First result word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
B	First base word	REAL	2
E	First exponent word	REAL	2
R	First result word	REAL	2

● Operand Specifications

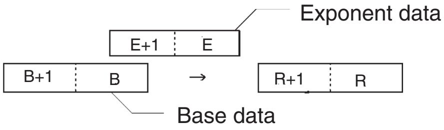
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B, E	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R											---							

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. • OFF in all other cases.
Underflow Flag	UF	• ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value. • OFF in all other cases.
Negative Flag	N	• ON if the result is negative. • OFF in all other cases.

Function

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 XY (X = B+1 and B; Y = E+1 and E).



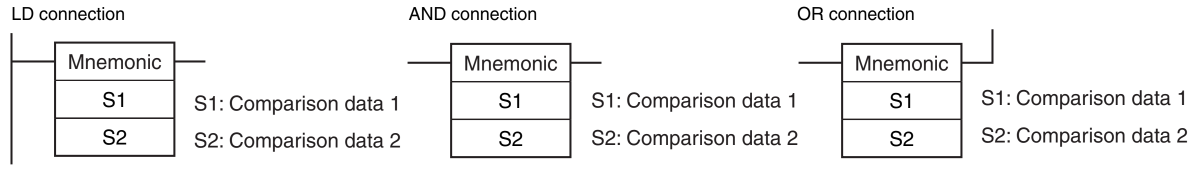
For example, when the base words (B+1 and B) contain 3.1 and the exponent words (E+1 and E) contain 3, the result is 3.1^3 or 29.791.

=F, <>F, <F, <=F, >F, >=F

Instruction	Mnemonic	Variations	Function code	Function
Single-precision Floating-point Comparison	=F <>F <F <=F >F >=F	---	329 330 331 332 333 334	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.

Symbol

Single-precision Floating-point Comparison



S1: Comparison data 1
S2: Comparison data 2

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S1	Comparison data 1	REAL	2
S2	Comparison data 2	REAL	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1 S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	OFF
Greater Than Flag	P_GT	• ON if S1+1, S1 > S2+1, S2. • OFF in all other cases.
Greater Than or Equal Flag	P_GE	• ON if S1+1, S1 ≥ S2+1, S2. • OFF in all other cases.
Equal Flag	P_EQ	• ON if S1+1, S1 = S2+1, S2. • OFF in all other cases.
Not Equal Flag	P_NE	• ON if S1+1, S1 ≠ S2+1, S2. • OFF in all other cases.
Less Than Flag	P_LT	• ON if S1+1, S1 < S2+1, S2. • OFF in all other cases.
Less Than or Equal Flag	P_LE	• ON if S1+1, S1 ≤ S2+1, S2. • OFF in all other cases.
Negative Flag	P_N	Unchanged

Function

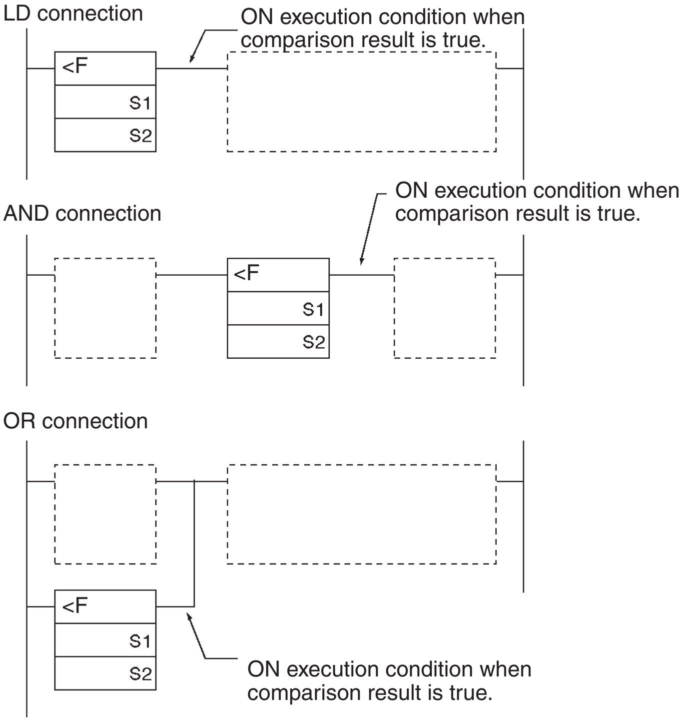
The input comparison instruction compares the data specified in S1 and S2 as single-precision floating point values (32-bit IEEE754 data) and creates an ON execution condition when the comparison condition is true.
When the data is stored in words, S1 and S2 specify the first of two words containing the 32-bit data. It is also possible to input the floating-point data as an 8-digit hexadecimal constant.

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

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 (LD, AND, and OR cannot be used in a ladder program)		Option (data format)
LD=, AND=, OR=, LD<>, AND<>, OR<>, LD<, AND<, OR<, LD<=, AND<=, OR<=, LD>, AND>, OR>, LD>=, AND>=, OR>=	+	F: Single-precision floating-point data

Code	Mnemonic	Name	Function
329	LD=F	LOAD FLOATING EQUAL	True if C1 = C2
	AND=F	AND FLOATING EQUAL	
	OR=F	OR FLOATING EQUAL	
330	LD<>F	LOAD FLOATING NOT EQUAL	True if C1 ≠ C2
	AND<>F	AND FLOATING NOT EQUAL	
	OR<>F	OR FLOATING NOT EQUAL	
331	LD<F	LOAD FLOATING LESS THAN	True if C1 < C2
	AND<F	AND FLOATING LESS THAN	
	OR<F	OR FLOATING LESS THAN	
332	LD<=F	LOAD FLOATING LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=F	AND FLOATING LESS THAN OR EQUAL	
	OR<=F	OR FLOATING LESS THAN OR EQUAL	
333	LD>F	LOAD FLOATING GREATER THAN	True if C1 > C2
	AND>F	AND FLOATING GREATER THAN	
	OR>F	OR FLOATING GREATER THAN	

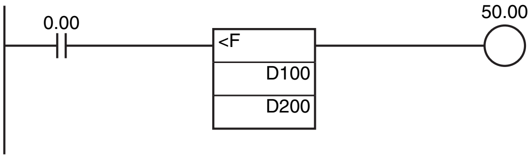
Code	Mnemonic	Name	Function
325	LD>=F	LOAD FLOATING GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=F	AND FLOATING GREATER THAN OR EQUAL	
	OR>=F	OR FLOATING GREATER THAN OR EQUAL	

Precautions

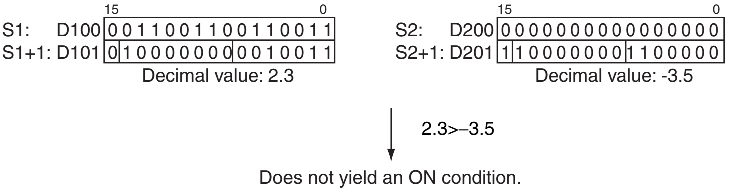
- Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

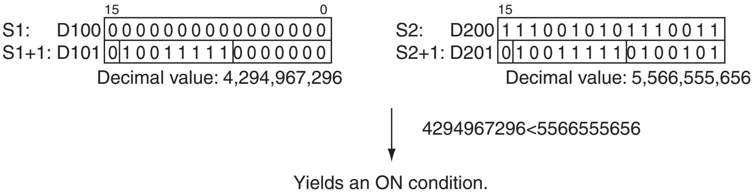
Example Programming

When CIO 0.00 is ON in the following example, the floating point data in D101, D100 is compared to the floating point data in D201, D200. If the content of D101, D100 is less than that of D201, D200, execution proceeds to the next line and CIO 50.00 is turned ON. If the content of D101, D100 is not less than that of D201, D200, execution does not proceed to the next instruction line.



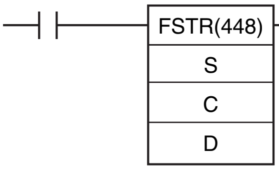
SINGLE FLOATING LESS THAN Comparison (<F)





FSTR

Instruction	Mnemonic	Variations	Function code	Function
FLOATING-POINT TO ASCII	FSTR	@FSTR	448	Expresses a 32-bit floating-point value (IEEE754-format) in standard decimal notation or scientific notation and converts that value to ASCII text.

	FSTR
Symbol	 S: First source word C: First control word D: First destination word

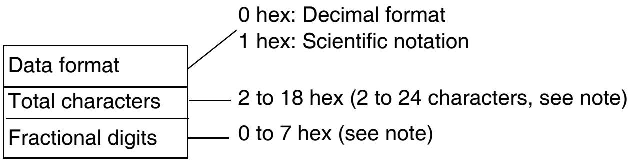
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
C	First control word	UINT	3
D	First destination word	UINT	Variable

C: FIrst Control Word



Note There are limits on the total number of characters and the number of fractional digits.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
C											---							
D																		

Flags

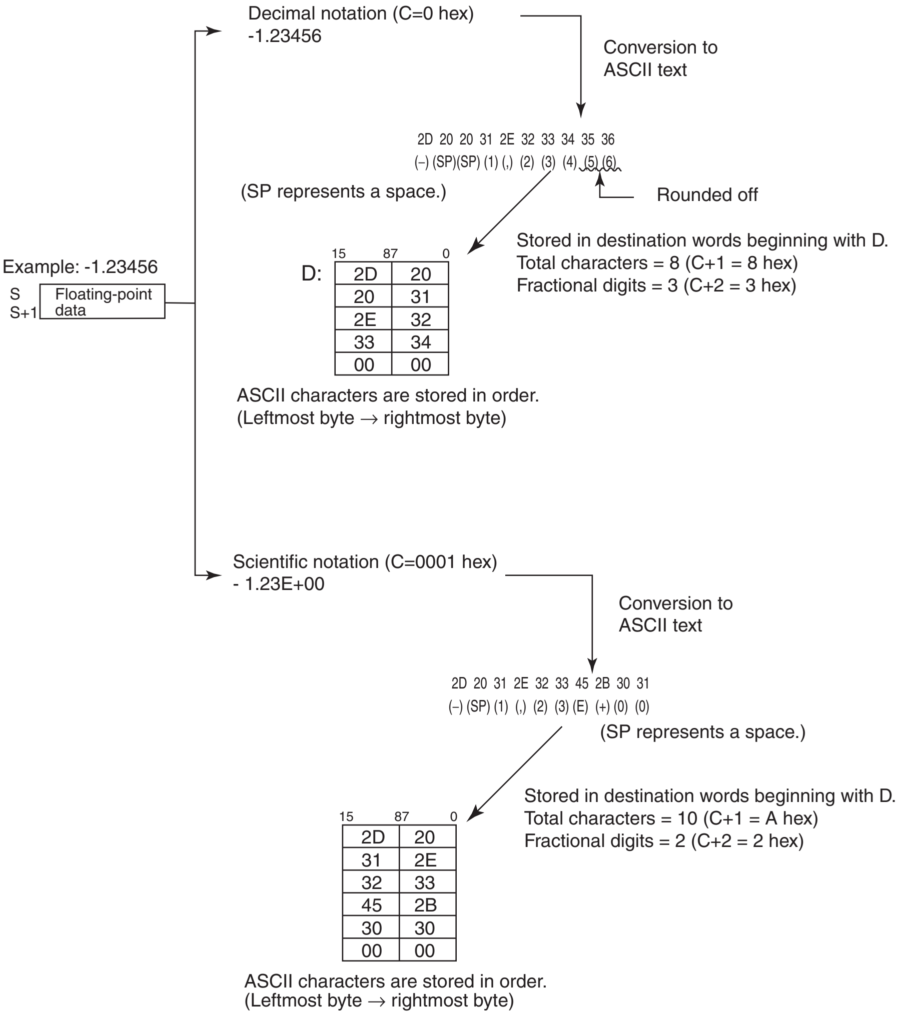
Name	Label	Operation
Error Flag	P_ER	• ON if the data in S+1 and S is not a valid floating-point number (NaN). • ON if the data in S+1 and S is +o or -o. • ON if the Data Format setting in C is not 0000 or 0001. • ON if the Total Characters setting in C+1 is not within the allowed range. (See 1. Limits 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	P_EQ	• ON if the conversion result is 0. • OFF in all other cases.

Function

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 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.

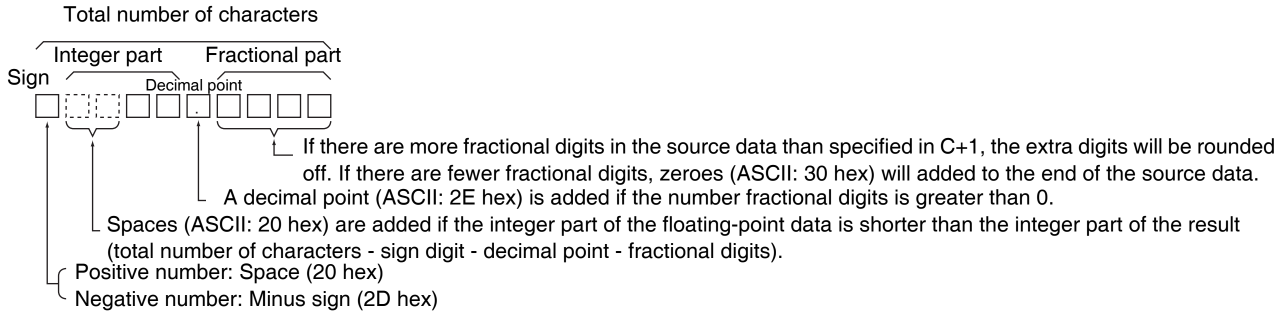
The ASCII text is stored in D and subsequent words in the following order: leftmost byte of D, rightmost byte of D, leftmost byte of D+1, rightmost byte of D+1, etc.



● 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=0 hex)



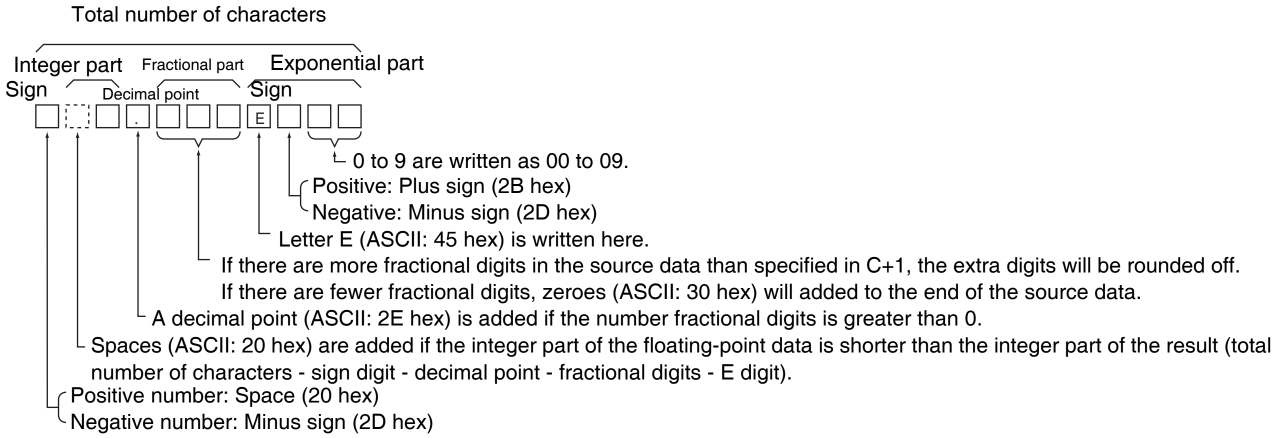
If there are more fractional digits in the source data than specified in C+1, the extra digits will be rounded off. If there are fewer fractional digits, zeroes (ASCII: 30 hex) will added to the end of the source data.

A decimal point (ASCII: 2E hex) is added if the number fractional digits is greater than 0.

Spaces (ASCII: 20 hex) are added if the integer part of the floating-point data is shorter than the integer part of the result (total number of characters - sign digit - decimal point - fractional digits).

Positive number: Space (20 hex)
Negative number: Minus sign (2D hex)

Scientific notation (C=1 hex)



0 to 9 are written as 00 to 09.

Positive: Plus sign (2B hex)
Negative: Minus sign (2D hex)

Letter E (ASCII: 45 hex) is written here.

If there are more fractional digits in the source data than specified in C+1, the extra digits will be rounded off. If there are fewer fractional digits, zeroes (ASCII: 30 hex) will added to the end of the source data.

A decimal point (ASCII: 2E hex) is added if the number fractional digits is greater than 0.

Spaces (ASCII: 20 hex) are added if the integer part of the floating-point data is shorter than the integer part of the result (total number of characters - sign digit - decimal point - fractional digits - E digit).

Positive number: Space (20 hex)
Negative number: Minus sign (2D 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.

- Limits on the Total Number of ASCII Characters
  1) Decimal Notation (C = 0 hex)
    - When there is no fractional part (C+2 = 0 hex):
      $2 \leq$ Total Characters ≤ 24
    - When there is a fractional part (C+2 = 1 to 7 hex):
      (Fractional digits + 3) $\leq$ Total Characters ≤ 24
  2) Scientific Notation (C = 1 hex)
    - When there is no fractional part (C+2 = 0 hex):
      $6 \leq$ Total Characters ≤ 24
    - When there is a fractional part (C+2 = 1 to 7 hex):
      (Fractional digits + 7) $\leq$ Total Characters ≤ 24

- Limits on the Number of Digits in the Integer Part

  1) Decimal Notation (C = 0 hex)

  - When there is no fractional part (C+2 = 0 hex):
    1 ≤ Number of Integer Digits ≤ 24
  - When there is a fractional part (C+2 = 1 to 7 hex):
    1 ≤ Number of Integer Digits ≤ (24 - Fractional digits - 2)

  2) Scientific Notation (C = 1 hex)
  1 digit (fixed)

- Limits on the Number of Digits in the Fractional Part

  1) Decimal Notation (C = 0 hex)

  - Fractional Digits ≤ 7
  - Also: Fractional Digits ≤ (Total Number of ASCII Characters - 3)

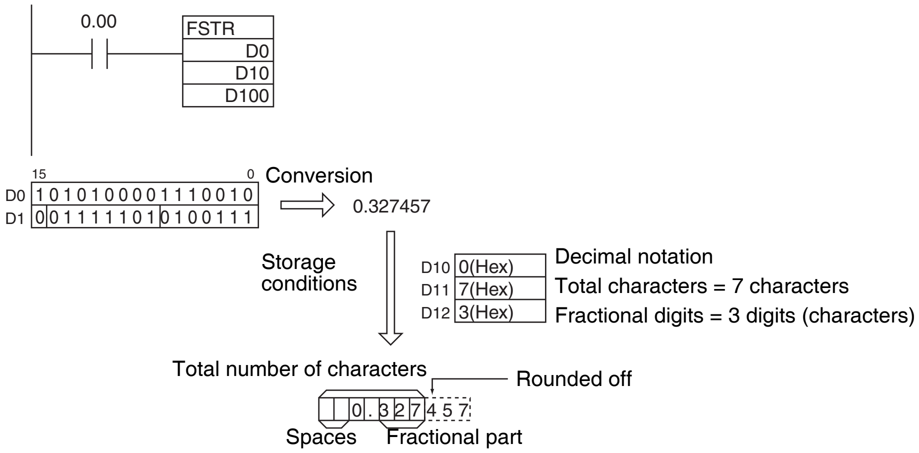
  2) Scientific Notation (C =1 hex)

  - Fractional Digits ≤ 7
  - Also: Fractional Digits ≤ (Total Number of ASCII Characters - 7)

Example Programming

● Converting to ASCII Text in Decimal Notation

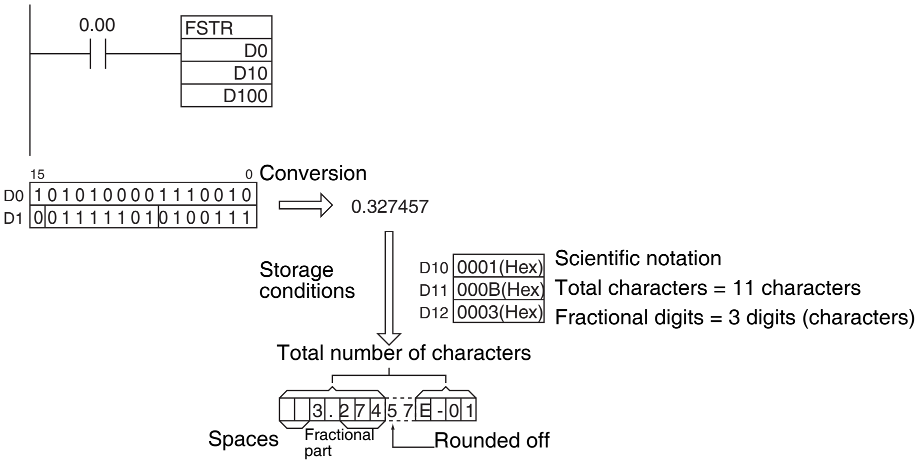
When CIO 0.00 is ON in the following example, FSTR(448) converts the floating-point data in D1 and D0 to decimal-notation ASCII text and writes the ASCII text to the destination words beginning with D100. The contents of the control words (D10 to D12) specify the details on the data format (decimal notation, 7 characters total, 3 fractional digits).



D100	20 (Space)	20 (Space)
D101	30 (0)	2E (.)
D102	33 (3)	32 (2)
D103	37 (7)	00

Converting to ASCII Text in Scientific Notation

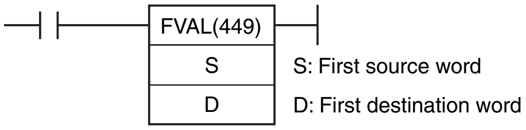
When CIO 0.00 is ON in the following example, FSTR(448) converts the floating-point data in D1 and D0 to scientific-notation ASCII text and writes the ASCII text to the destination words beginning with D100. The contents of the control words (D10 to D12) specify the details on the data format (scientific notation, 11 characters total, 3 fractional digits).



D100	20 (Space)	20 (Space)
D101	33 (3)	2E (.)
D102	32 (2)	37 (7)
D103	35 (5)	45 (E)
D104	2D (–)	30 (0)
D105	31 (1)	00

FVAL

Instruction	Mnemonic	Variations	Function code	Function
ASCII TO FLOATING-POINT	FVAL	@FVAL	449	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.

Symbol	FVAL
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	UINT	Variable
D	First destination word	REAL	2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

Flags

Name	Label	Operation
Error Flag	P_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-). 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 +o or -o after conversion. • ON if there 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	P_EQ	• ON if the conversion result is 0. • OFF in all other cases.

Function

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:

Up to 6 characters are valid, excluding the sign, decimal point, and exponent. Any characters beyond the 6th character will be ignored.

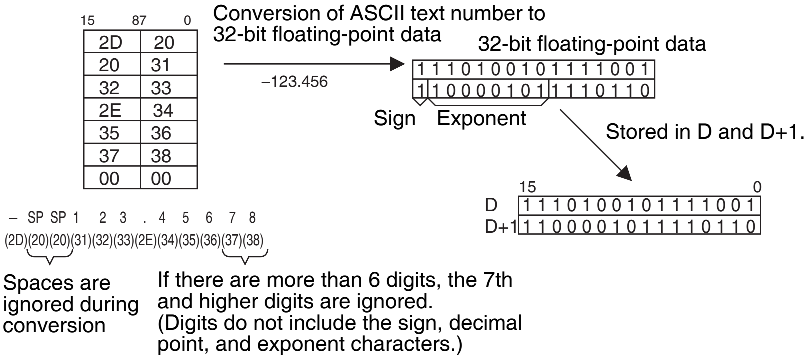
- Decimal Notation
  Real numbers expressed with an integer and fractional part.
  Example: 124.56

- Scientific Notation
  Real numbers expressed as an integer part, fractional part, and exponent part.
  Example: 1.2456E-2 (1.2456×10^{-2})

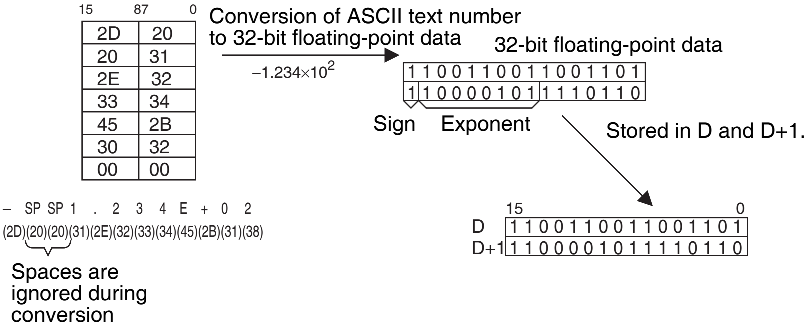
The data format (decimal or scientific notation) is detected automatically.

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



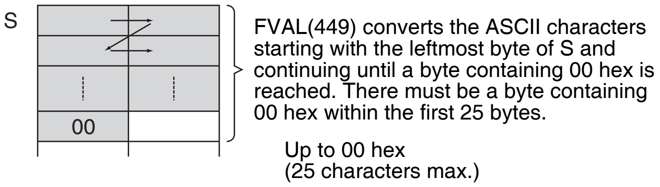
Scientific notation



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.

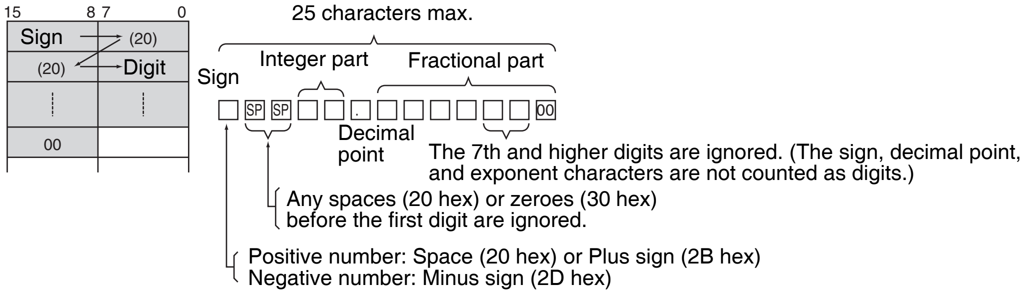
ASCII Character Storage



FVAL(449) converts the ASCII characters starting with the leftmost byte of S and continuing until a byte containing 00 hex is reached. There must be a byte containing 00 hex within the first 25 bytes.

Up to 00 hex
(25 characters max.)

Decimal notation

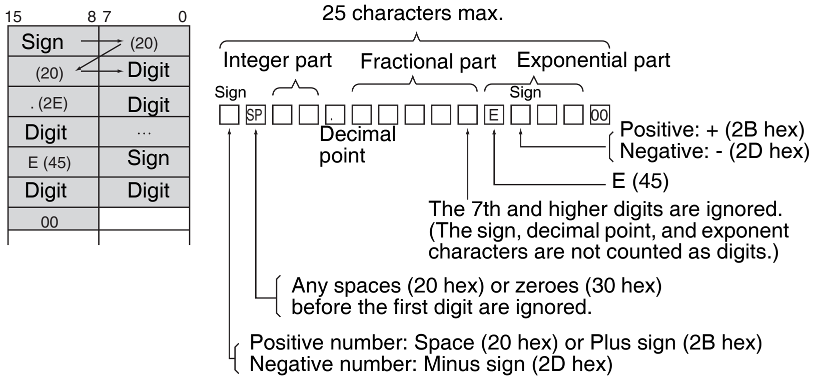


The 7th and higher digits are ignored. (The sign, decimal point, and exponent characters are not counted as digits.)

Any spaces (20 hex) or zeroes (30 hex) before the first digit are ignored.

Positive number: Space (20 hex) or Plus sign (2B hex)
Negative number: Minus sign (2D hex)

Scientific notation



Positive: + (2B hex)
Negative: - (2D hex)

E (45)

The 7th and higher digits are ignored. (The sign, decimal point, and exponent characters are not counted as digits.)

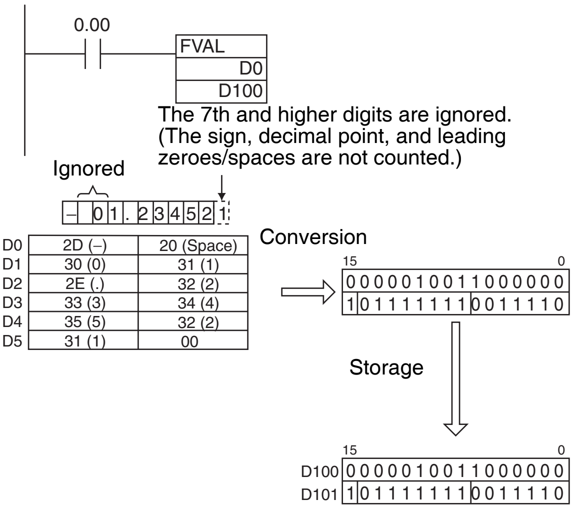
Any spaces (20 hex) or zeroes (30 hex) before the first digit are ignored.

Positive number: Space (20 hex) or Plus sign (2B hex)
Negative number: Minus sign (2D hex)

Example Programming

● 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.

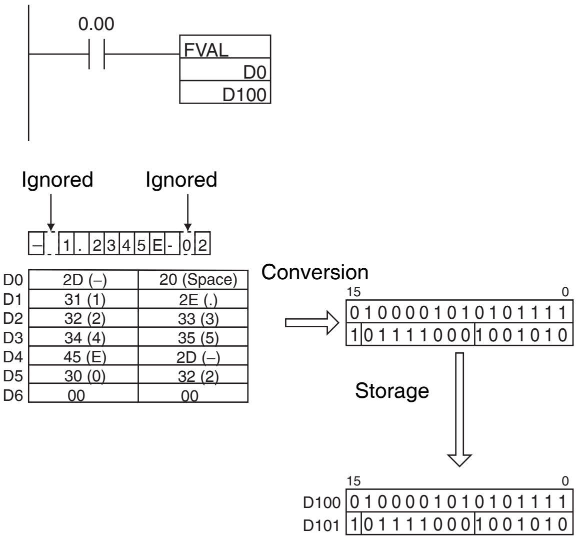


The 7th and higher digits are ignored. (The sign, decimal point, and leading zeroes/spaces are not counted.)

D0	2D (–)	20 (Space)
D1	30 (0)	31 (1)
D2	2E (.)	32 (2)
D3	33 (3)	34 (4)
D4	35 (5)	32 (2)
D5	31 (1)	00

● Converting ASCII Text in Scientific Notation

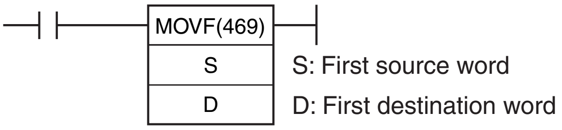
When CIO 0.00 is ON in the following example, FVAL(449) converts the specified scientific-notation ASCII text number in the source words starting at D0 to floating-point data and writes the result to destination words D100 and D101.



D0	2D (–)	20 (Space)
D1	31 (1)	2E (.)
D2	32 (2)	33 (3)
D3	34 (4)	35 (5)
D4	45 (E)	2D (–)
D5	30 (0)	32 (2)
D6	00	00

MOVF

Instruction	Mnemonic	Variations	Function code	Function
MOVE FLOATING-POINT (SINGLE)	MOVF	@MOVF	469	Transfers the specified 32-bit floating-point number to the destination words.

	MOVF
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	REAL	2
D	First destination word	REAL	2

● Operand Specifications

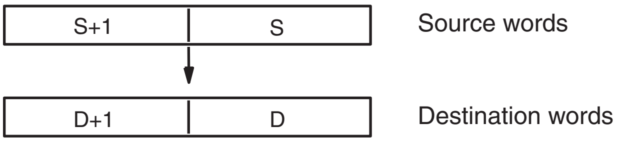
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
D											---							

Flags

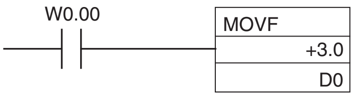
Name	Label	Operation
Error Flag	P_ER	OFF
Equals Flag	P_EQ	• ON if the source data is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if the source data is negative. • OFF in all other cases.

Function

MOVF(469) outputs the single-precision floating-point number (32-bit source data in IEEE754 format) from source words S+1 and S to destination words D+1 and D.



Example Programming



When input condition W0.00 is ON, the content of D0 and D1 (+3.0) is stored in floating-point format (IEEE754 format).

D1	D0
#4040	#0000

Equivalent to +3.0 in floating-point format

Double-precision Floating-point Instructions

Double-precision Floating-point Instructions

■ Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

Real number = $(-1)^{s}\ 2^{e-1,023}\ (1.f)$

s: Sign

e: Exponent

f: Mantissa

The floating-point data format conforms to the IEEE754 standards. Data is expressed in 32 bits, as follows:

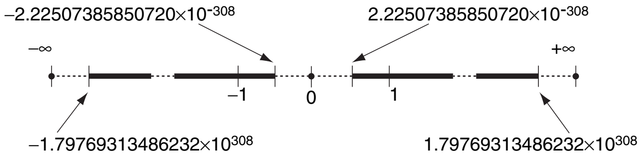
Data	No. of bits	Contents
s: sign	1	0: positive; 1: negative
e: exponent	11	The exponent (e) value ranges from 0 to 2,047. The actual exponent is the value remaining after 1,023 is subtracted from e, resulting in a range of -1,023 to 1,024. “e=0” and “e=2,047” express special numbers.
f: mantissa	52	The mantissa portion of binary floating-point data fits the format $2.0 > 1.f \geq 1.0$.

■ Number of Digits

Fifteen digits are effective for double-precision floating-point data.

■ Floating-point Data

- The following data can be expressed by floating-point data:
  - -∞
  - $-1.79769313486232 \times 10^{308} \leq \text{value} \leq -2.22507385850720 \times 10^{-308}$
  - 0
  - $2.22507385850720 \times 10^{-308} \leq \text{value} \leq 1.79769313486232 \times 10^{30}$
  - +∞
  - Not a number (NaN)

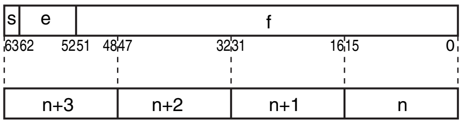


- Special Numbers
  The formats for NaN, ±∞, and 0 are as follows:
  - NaN*:e = 2,047 and f ≠ 0
  - +∞:e = 2,047, f = 0, and s= 0
  - -∞: e = 2,047, f = 0, and s= 1
  - 0:   e = 0 and f = 0

* 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.

(1) Normalized Numbers

Normalized numbers express real numbers. The sign bit will be 0 for a positive number and 1 for a negative number.

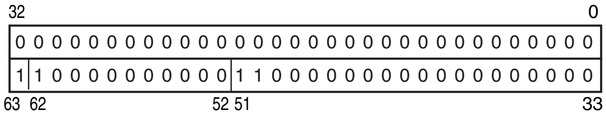
The exponent (e) will be expressed from 1 to 2,046, and the real exponent will be 1,023 less, i.e., -1,022 to 1,023.

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: $-1.75 \times 2^{1} = -3.5$

(2) 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.

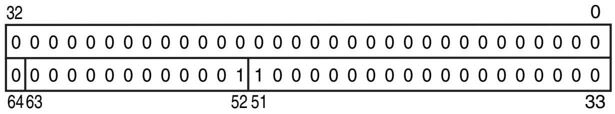
The exponent (e) will be 0, and the real exponent will be -1,022.

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: $-0.75 \times 2^{-1,022} = 1.668805 \times 10^{-308}$

(3) 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.

(4) Infinity

Values of $+\infty$ and $-\infty$ 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.

(5) NaN

NaN (not a number) is produced when the result of calculations, such as 0.0/0.0, ∞/∞, or $\infty-\infty$, 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

(1) 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.

(2) 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.

(3) 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

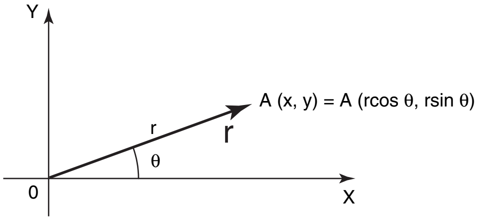
- When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞. If the result is positive, it will be output as +∞; if negative, then -∞.
- The Equals Flag will only turn ON when 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

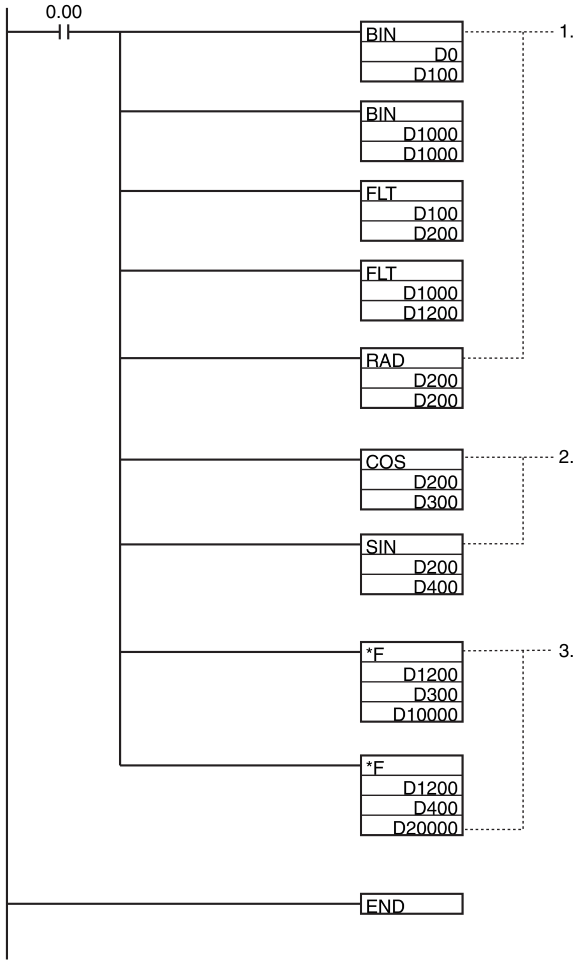
This example shows the differences in between single-precision and double-precision calculations when the following vector expressed in polar coordinates is converted to rectangular coordinates A (x,y).

$$r = re^{j\left(\frac{\pi}{360}\right)\theta}$$

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.



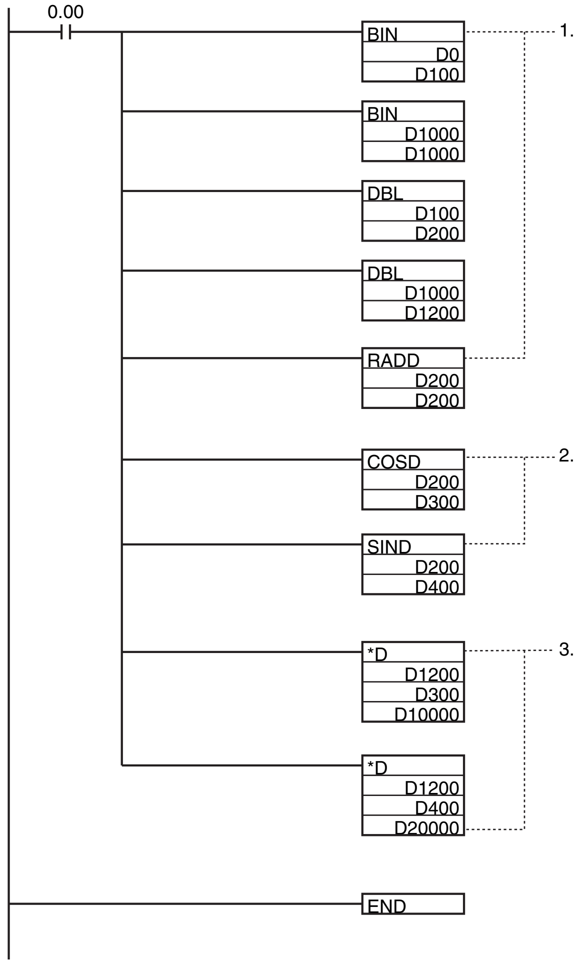
■ Ladder Program for the Single-precision Calculation



1. This program section converts the BCD data to single-precision floating-point data (32 bits, IEEE754-format).
   - 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.
   - The floating-point data for the angle θ is output to D200 and D201.
   - RAD(458) converts the angle data in D200 and D201 to radians.
   - 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.
   - The value for cos θ is output to D300 and D301.
   - 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).
   - The value for x (r × cos θ) is output to D10000 and D10001.
   - 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

■ Ladder Program for the Double-precision Calculation



1. This program section converts the BCD data to double-precision floating-point data (64 bits, IEEE754-format).
   - 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.
   - The floating-point data for the angle θ is output to D200 and D203.
   - RADD(849) converts the angle data in words D200 and D203 to radians.
   - The floating-point data for the radius r is output to words D1200 and D1203.
2. This program section calculates the sin θ and the cos θ as double-precision floating-point values.
   - The value for cos θ is output to words D300 and D303.
   - 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).
   - The value for x (r × cos θ) is output to words D10000 and D10003.
   - The value for y (r × sin θ) is output to D20000 and D20003.

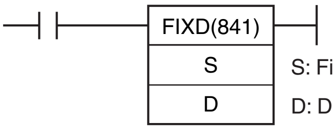
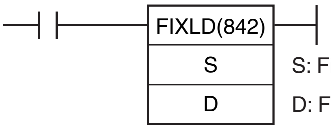
Coordinate	Floating-point number	Real number
x	4022 CB39 E973 5C32	3.4202014332567
y	400B 5C92 91AC 8EEB	9.3969262078591

■ 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.

FIXD/FIXLD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE FLOATING TO 16-BIT	FIXD	@FIXD	841	Converts a double-precision (64-bit) floating-point value to 16-bit signed binary data and places the result in the specified result word.
DOUBLE FLOATING TO 32-BIT	FIXLD	@FIXLD	842	Converts a double-precision (64-bit) floating-point value to 32-bit signed binary data and places the result in the specified result words.

	FIXD	FIXLD
Symbol	 S: First source word D: Destination word	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		FIXD	FIXLD	FIXD	FIXLD
S	First source word	LREAL	LREAL	4	4
D	Destination word	INT	DINT	1	2

■ Operand Specifications

- FIXD

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D												OK						

- FIXLD

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

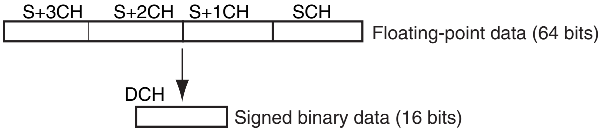
Flags

Name	Label	Operation	
		FIXD	**FIXLD**
Error Flag	P_ER	• ON if the source data is not recognized as floating-point data. • ON if the source data (S) is not a number (NaN). • ON if the integer portion of the source data (S) is not within the range of -32,768 to 32,767. • OFF in all other cases.	• ON if the source data is not recognized as floating-point data. • 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	P_EQ	• ON if the result is 0000/0000 0000. • OFF in all other cases.	
Negative Flag	P_N	• ON if bit 15 of the result is ON. • OFF in all other cases.	

Function

■ FIXD

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.

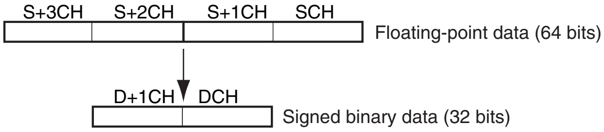
Example conversions:

A floating-point value of 3.5 is converted to 3.

A floating-point value of -3.5 is converted to -3.

■ FIXD

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.

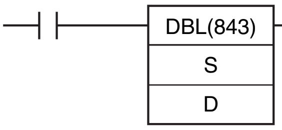
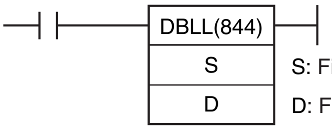
Example conversions:

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.

DBL/DBLL

Instruction	Mnemonic	Variations	Function code	Function
16-BIT TO DOUBLE FLOATING	DBL	@DBL	843	Converts a 16-bit signed binary value to double-precision (64-bit) floating-point data and places the result in the specified destination words.
32-BIT TO DOUBLE FLOATING	DBLL	@DBLL	844	Converts a double-precision (64-bit) floating-point value to 32-bit signed binary data and places the result in the specified result words.

	DBL	DBLL
Symbol	 DBL(843) S D S: Source word D: First destination word	 DBLL(844) S D S: First Source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type		Size	
		DBL	DBLL	DBL	DBLL
S	DBL: Source word DBLL: First source word	INT	DINT	1	2
D	First destination word	LREAL	LREAL	4	4

■ Operand Specifications

- DBL

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
D											---	---						

- DBLL

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
D											---							

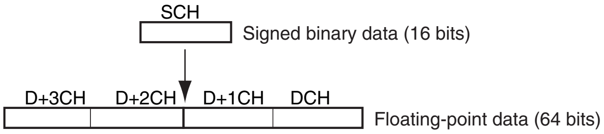
Flags

Name	Label	Operation
Error Flag	P_ER	OFF
Equals Flag	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

Function

■ DBL

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).

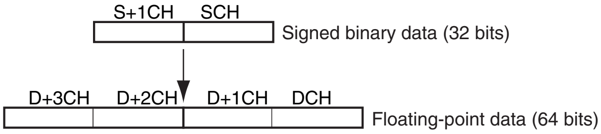
Example conversions:

A signed binary value of 3 is converted to 3.0.

A signed binary value of -3 is converted to -3.0.

■ DBLL

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.

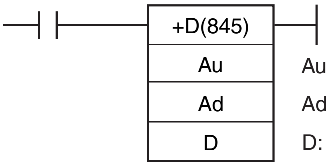
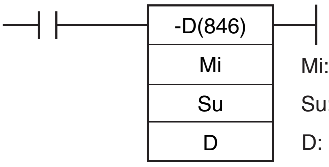
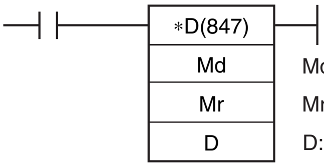
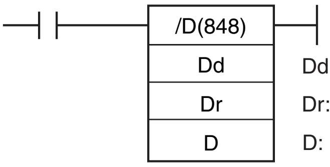
Example Conversions:

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.

+D, -D, ∗D, /D

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE FLOATING-POINT ADD	+D	@+D	845	Adds two double-precision (64-bit) floating-point numbers and places the result in the specified destination words.
DOUBLE FLOATING-POINT SUBTRACT	–D	@-D	846	Subtracts one double-precision (64-bit) floating-point number from another and places the result in the specified destination words.
DOUBLE FLOATING-POINT MULTIPLY	∗D	@∗D	847	Multiplies two double-precision (64-bit) floating-point numbers and places the result in the specified result words.
DOUBLE FLOATING-POINT DIVIDE	/D	@/D	848	Divides one double-precision (64-bit) floating-point number by another and places the result in the specified destination words.

	+D	-D
Symbol	 Au: First augend word Ad: First addend word D: First destination word	 Mi: First Minuend word Su: First Subtrahend word D: First destination word
	∗D	/D
	 Md: First Multiplicand wor Mr: First Multiplier word D: First destination word	 Dd: First Dividend word Dr: First Divisor word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
Au	+D: First augend word	LREAL	4
Mi	–D: First Minuend word		
Md	∗D: First Multiplicand word		
Dd	/D: First Dividend word		
Ad	+D: First addend word	LREAL	4
Su	–D: First Subtrahend word		
Mr	∗D: First Multiplier word		
Dr	/D: First Divisor word		
D	First destination word	LREAL	4

■ Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CI O	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
Au, Mi, Md, Dd, Ad, Su, Mr, Dr	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

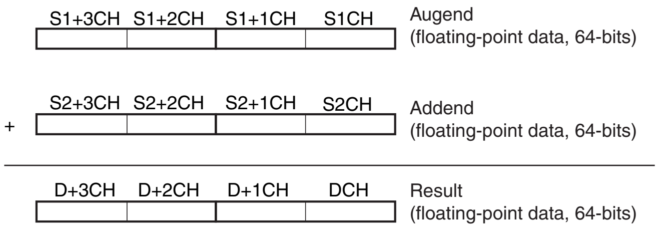
Flags

Name	Label	Operation
Error Flag	P_ER	• ON when S1 or S2 is not treated as floating-point data. • +D •ON if the augend or addend data is not a number (NaN). •ON if +o is added to -o. • -D •ON if the minuend or subtrahend is not a number (NaN). •ON if +o is subtracted from +o. •ON if -o is subtracted from -o. • *D •ON if the multiplicand or multiplier is not a number (NaN). •ON if +o and 0 are multiplied. •ON if -o and 0 are multiplied. • /D •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 +o or -o. • OFF in all other cases.
Equals Flag	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	P_UF	• ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

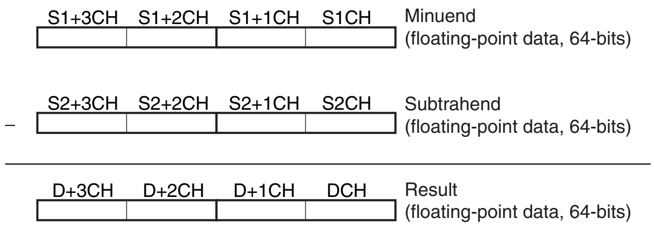
Function

The data specified in S1 and the data specified in S2 are added (+D(845)), subtracted (-D(846)), multiplied (*D(847)), or divided (/D(848)) as double-precision floating-point data (64 bits: IEEE754) and the result is output to D+3, D+2, D+1, and D.

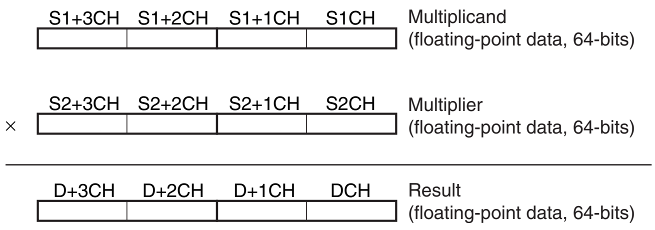
- +D



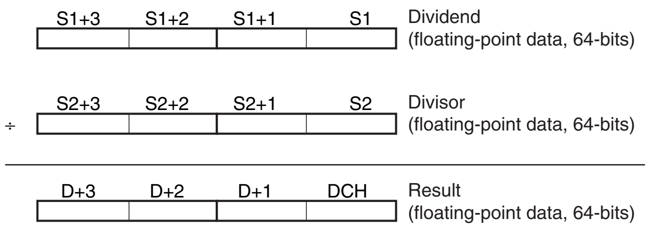
- -D



- *D



- /D



■ Operation rules

The result of an operation is output as shown below depending on the combination of floating-point data.

● DOUBLE FLOATING-POINT ADD (+D)

		Augend				
Addend		**0**	**Numeral**	**+∞**	**−∞**	**NaN**
	0	0	Numeral	+∞	−∞	
	Numeral	Numeral	**	+∞	−∞	
	+∞	+∞	+∞	+∞	ER	
	−∞	−∞	−∞	ER	−∞	
	NaN					ER

* *: The results could be zero (including underflows), a numeral, +o, or -o.
ER: The Error Flag will be turned ON and the instruction will not be executed.

● DOUBLE FLOATING-POINT SUBTRACT (-D)

		Minuend				
Subtrahend		**0**	**Numeral**	**+∞**	**−∞**	**NaN**
	0	0	Numeral	+∞	−∞	
	Numeral	Numeral	**	+∞	−∞	
	+∞	−∞	−∞	ER	−∞	
	−∞	+∞	+∞	+∞	ER	
	NaN					ER

* *: The results could be zero (including underflows), a numeral, +o, or -o.
ER: The Error Flag will be turned ON and the instruction will not be executed.

● DOUBLE FLOATING-POINT MULTIPLY (∗D)

		Multiplicand				
Multiplier		**0**	**Numeral**	**+∞**	**−∞**	**NaN**
	0	0	0	ER	ER	
	Numeral	0	**	+/−∞	+/−∞	
	+∞	ER	+/−∞	+∞	−∞	
	−∞	ER	+/−∞	−∞	+∞	
	NaN					ER

* *: The results could be zero (including underflows), a numeral, +o, or -o.
ER: The Error Flag will be turned ON and the instruction will not be executed.

● DOUBLE FLOATING-POINT DIVIDE (/D)

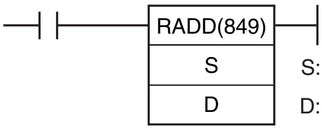
		Dividend				
Divisor		**0**	**Numeral**	**+∞**	**−∞**	**NaN**
	0	ER	+/−∞	+∞	−∞	
	Numeral	0	**	+/−∞	+/−∞	
	+∞	0	*	ER	ER	
	−∞	0	*	ER	ER	
	NaN					ER

* : The results will be zero for underflows.

* *: The results could be zero (including underflows), a numeral, +o, or -o.
ER: The Error Flag will be turned ON and the instruction will not be executed.

RADD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE DEGREES TO RADIANS	RADD	@RADD	849	Converts a double-precision (64-bit) floating-point number from degrees to radians and places the result in the specified result words.

	RADD
Symbol	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

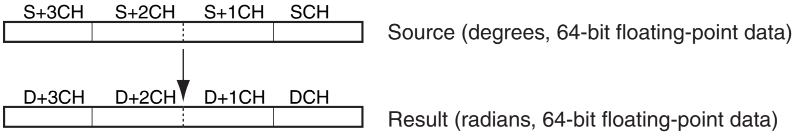
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the angle 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	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value. • OFF in all other cases.
Underflow Flag	P_UF	• ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value. • OFF in all other cases.
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

Function

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.)

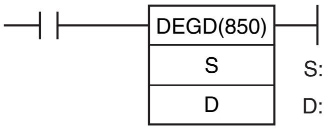


Degrees are converted to radians by means of the following formula:

Degrees × π/180 = radians

DEGD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE RADIANS TO DEGREES	DEGD	@DEGD	850	Converts a double-precision (64-bit) floating-point number from radians to degrees and places the result in the specified result words.

	DEGD
Symbol	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

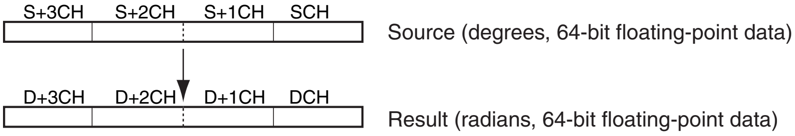
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the angle 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	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value. • OFF in all other cases.
Underflow Flag	P_UF	• ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value. • OFF in all other cases.
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

Function

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.)

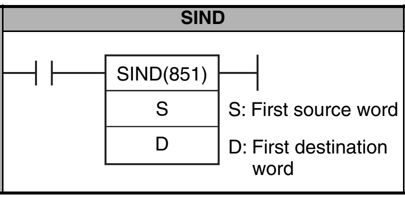
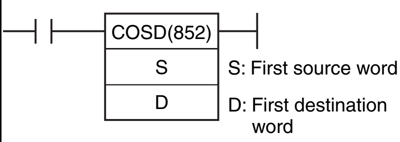
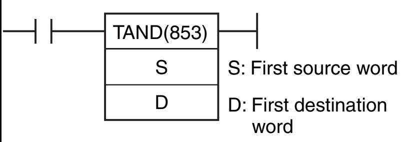


Radians are converted to degrees by means of the following formula:

Radians × π180/p = degrees

SIND/COSD/TAND

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE SINE	SIND	@SIND	851	Calculates the sine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.
DOUBLE COSINE	COSD	@COSD	852	Calculates the cosine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.
DOUBLE TANGENT	TAND	@TAND	853	Calculates the tangent of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

	SIND	COSD	TAND
Symbol	 SIND(851) S D S: First source word D: First destination word	 COSD(852) S D S: First source word D: First destination word	 TAND(853) S D S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

Flags

Name	Label	Operation
Error Flag	P_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 65,535. • OFF in all other cases.
Equals Flag	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• SIND, COSD Unchanged • TAND •ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value. •OFF in all other cases.
Underflow Flag	P_UF	Unchanged
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

Function

■ SIND

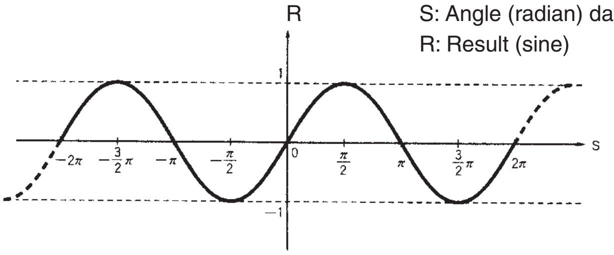
SIND(851) calculates the sine of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.

(The floating point source data must be in IEEE754 format.)

SIN(S+3 S+2 S+1 S) → D+3 D+2 D+1 D

- Specify the desired angle (-65,535 to 65,535) in radians in words S to S+3.
  For information on converting between degrees and radians, see 3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849) or 3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850).
- If the angle is outside of the range -65,535 to 65,535, an error will occur and the instruction will not be executed.

● The following diagram shows the relationship between the angle and result.



S: Angle (radian) data
R: Result (sine)

■ COSD

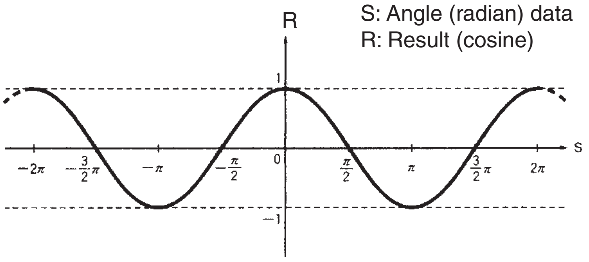
COSD(852) calculates the cosine of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.

(The floating point source data must be in IEEE754 format.)

COS(S+3 S+2 S+1 S) → D+3 D+2 D+1 D

- Specify the desired angle (-65,535 to 65,535) in radians in words S to S+3.
  For information on converting between degrees and radians, see 3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849) or 3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850).
- If the angle is outside of the range -65,535 to 65,535, an error will occur and the instruction will not be executed.

● The following diagram shows the relationship between the angle and result.



S: Angle (radian) data
R: Result (cosine)

■ TAND

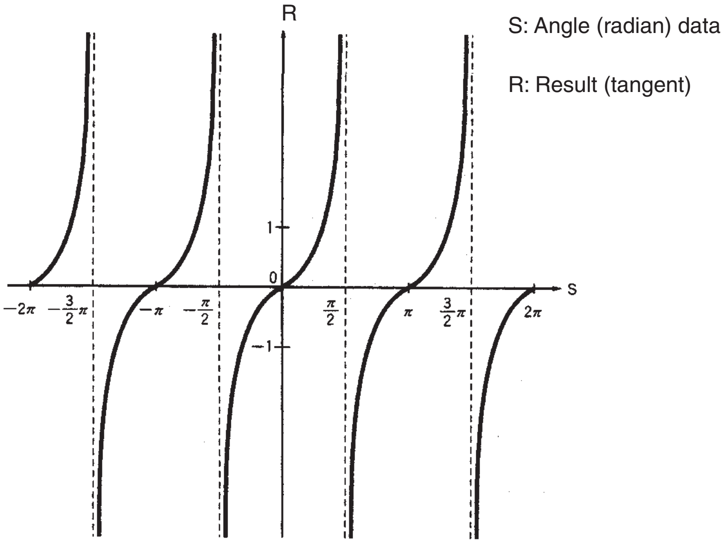
TAND(853) calculates the tangent of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.

(The floating point source data must be in IEEE754 format.)

TAN(S+3 S+2 S+1 S) → D+3 D+2 D+1 D

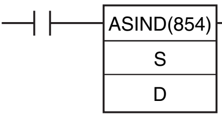
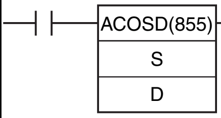
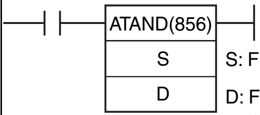
- Specify the desired angle (-65,535 to 65,535) in radians in words S to S+3.
  For information on converting between degrees and radians, see 3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849) or 3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850).
- If the angle is outside of the range -65,535 to 65,535, an error 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 angle and result.



ASIND/ACOSD/ATAND

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE ARC SINE	ASIND	@ASIND	854	Calculates the arc sine of a double-precision (64-bit) floating-point number and places the result in the specified destination words.
DOUBLE ARC COSINE	ACOSD	@ACOSD	855	Calculates the arc cosine of a double-precision (64-bit) floating-point number and places the result in the specified result words.
DOUBLE ARC TANGENT	ATAND	@ATAND	856	Calculates the arc tangent of a double-precision (64-bit) floating-point number and places the result in the specified result words.

	ASIND	ACOSD	ATAND
Symbol	 S: First source word D: First destination word	 S: First source word D: First destination word	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the source data is not recognized as floating-point data. • ASIND •ON if the source data is not a number (NaN). •ON if the absolute value of the source data exceeds 1.0. • ACOSD •ON if the source data is not a number (NaN). •ON if the absolute value of the source data exceeds 1.0. • ATAND •ON if the source data is not a number (NaN). •OFF in all other cases.
Equals Flag	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	Unchanged
Underflow Flag	P_UF	Unchanged
Negative Flag	P_N	• ASIND, ATAND •ON if the result is negative. •OFF in all other cases. • ACOSD Unchanged

Function

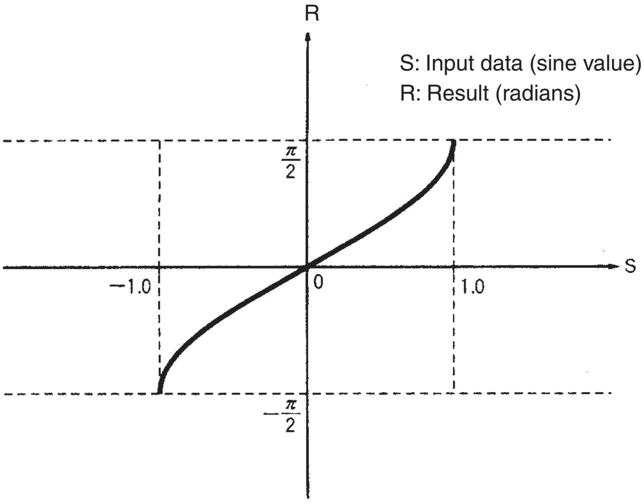
■ ASIND

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.)

SIN^{-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 -π/2 to π/2.

● The following diagram shows the relationship between the input data and result.



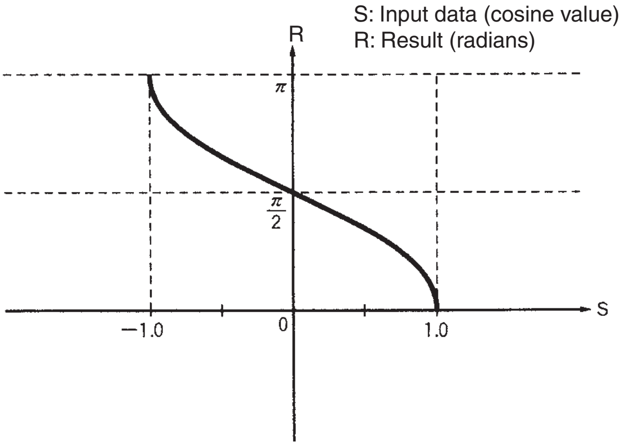
■ ACOSD

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 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.



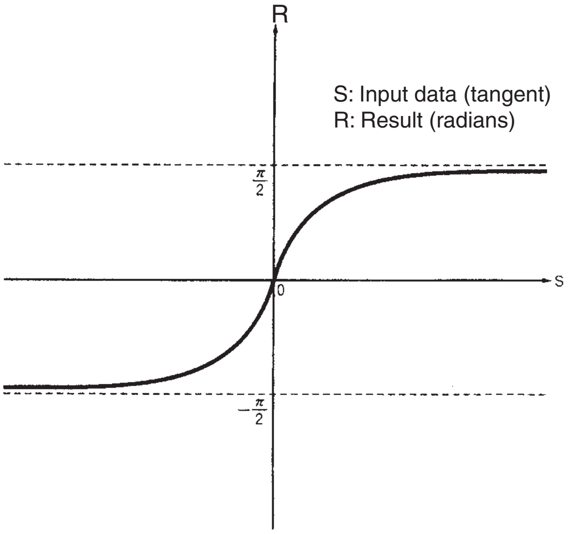
■ ATAND

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

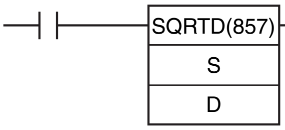
The result is output to words D to D+3 as an angle (in radians) within the range of -π/2 to π/2.

● The following diagram shows the relationship between the input data and result.



SQRTD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE SQUARE ROOT	SQRTD	@SQRTD	857	Calculates the square root of a double-precision (64-bit) floating-point number and places the result in the specified result words.

	SQRTD
Symbol	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

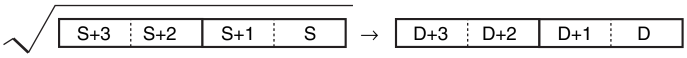
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

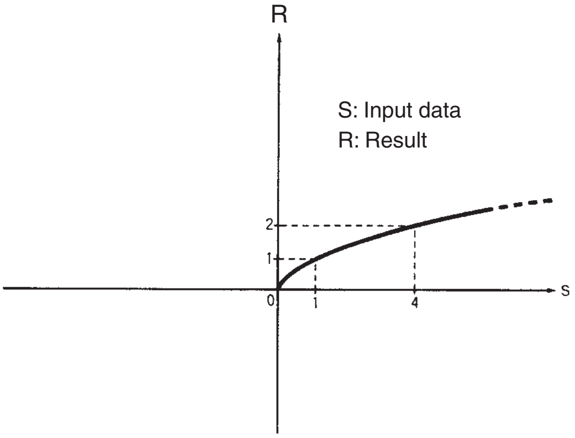
Name	Label	Operation
Error Flag	P_ER	• ON if the input data is not recognized as floating-point data. • ON if the source data is negative. • ON if the source data is not a number (NaN). • OFF in all other cases.
Equals Flag	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value. • OFF in all other cases.
Underflow Flag	P_UF	Unchanged
Negative Flag	P_N	Unchanged

Function

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.)

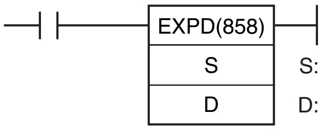


■ The following diagram shows the relationship between the input data and result.



EXPD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE EXPONENT	EXPD	@EXPD	858	Calculates the natural (base e) exponential of a double-precision (64-bit) floating-point number and places the result in the specified result words.

	EXPD
Symbol	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

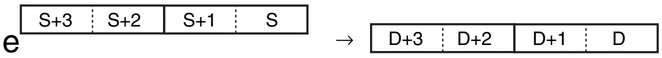
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the input 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	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value. • OFF in all other cases.
Underflow Flag	P_UF	• ON if the absolute value of the result is too small to be expressed as a double-precision (64-bit) floating-point value. • OFF in all other cases.
Negative Flag	P_N	Unchanged

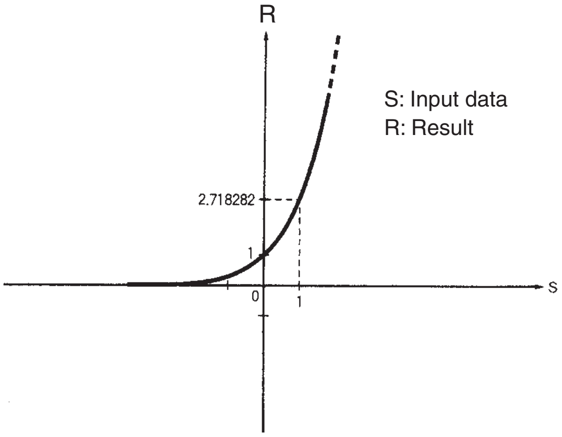
Function

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 ex (x = source) and places the result in words D to D+3.



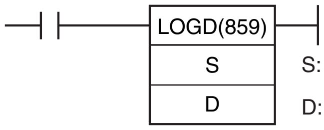
- The constant e is 2.718282.

■ The following diagram shows the relationship between the input data and result.



LOGD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE LOGARITHM	LOGD	@LOGD	859	Calculates the natural (base e) logarithm of a double-precision (64-bit) floating-point number and places the result in the specified destination words.

	LOGD
Symbol	 S: First source word D: First destination word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the input 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	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value. • OFF in all other cases.
Underflow Flag	P_UF	Unchanged
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

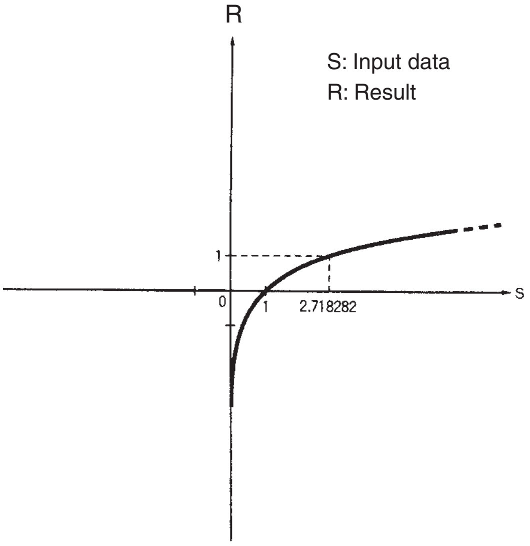
Function

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 |

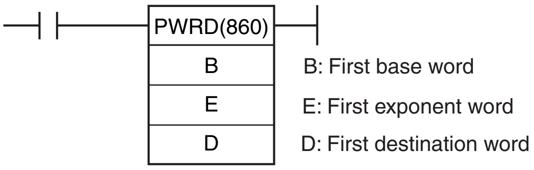
- The constant e is 2.718282.

■ The following diagram shows the relationship between the input data and result.



PWRD

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE EXPONENTIAL POWER	PWRD	@PWRD	860	Raises a double-precision (64-bit) floating-point number to the power of another double-precision (64-bit) floating-point number.

	PWRD
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
B	First base word	LREAL	4
E	First exponent word	LREAL	4
D	First destination word	LREAL	4

■ Operand Specifications

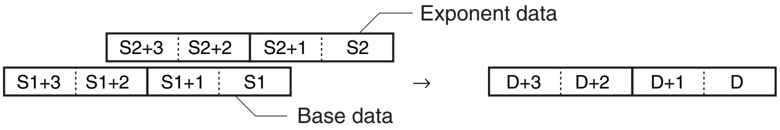
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B, E D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_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	P_EQ	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Overflow Flag	P_OF	• ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value. • OFF in all other cases.
Underflow Flag	P_UF	• ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value. • OFF in all other cases.
Negative Flag	P_N	• ON if the result is negative. • OFF in all other cases.

Function

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 XY (X = content of B to B+3; Y = content of E to E+3).



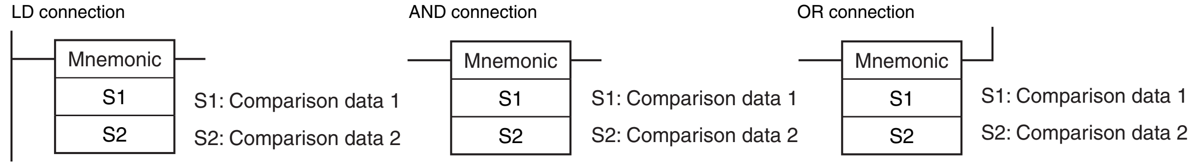
For example, when the base words (B to B+3) contain 3.1 and the exponent words (E to E+3) contain 3, the result is 3.13 or 29.791.

=D, <>D, <D, <=D, >D, >=D

Instruction	Mnemonic	Variations	Function code	Function
Double-precision Floating-point Input Comparison	=D <>D <D <=D >D >=D	---	335 336 337 338 339 340	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.

Symbol

Double-precision Floating-point Input Comparison



Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S1	Comparison data 1	LREAL	4
S2	Comparison data 2	LREAL	4

■ Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1 S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

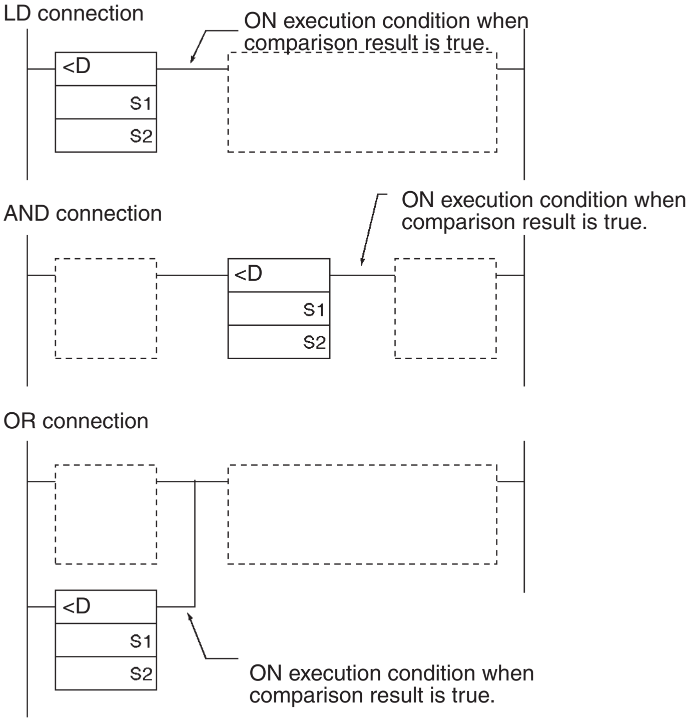
Name	Label	Operation
Error Flag	P_ER	OFF
Greater Than Flag	P_GT	• ON if both the exponent and mantissa of the result are 0. • OFF in all other cases.
Greater Than or Equal Flag	P_GE	• ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value. • OFF in all other cases.
Equal Flag	P_EQ	Unchanged
Not Equal Flag	P_NE	• ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value. • OFF in all other cases.
Less Than Flag	P_LT	Unchanged
Less Than or Equal Flag	P_LE	• ON if the result is negative. • OFF in all other cases.
Negative Flag	P_N	Unchanged

Function

The input comparison instruction compares the data specified in S1 and S2 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, S1 and S2 specify the first of four words containing the 64-bit data. The 64-bit floating-point data cannot be input as constants.

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

- 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 (LD, AND, and OR cannot be used in a ladder program)		Option (data format)
LD=, AND=, OR=, LD<>, AND<>, OR<>, LD<, AND<, OR<, LD<=, AND<=, OR<=LD>, AND>, OR>, LD>=, AND>=, OR>=	+	D: Double-precision floating-point data

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	
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	

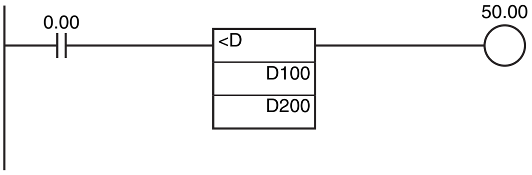
Code	Mnemonic	Name	Function
340	LD>=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	
	OR>=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	

Precautions

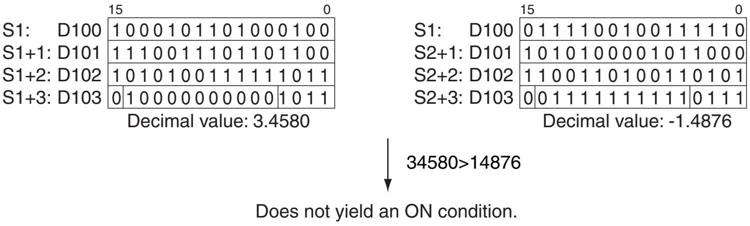
Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

Example Programming

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 50.00 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.



DOUBLE FLOATING LESS THAN Comparison (<D)

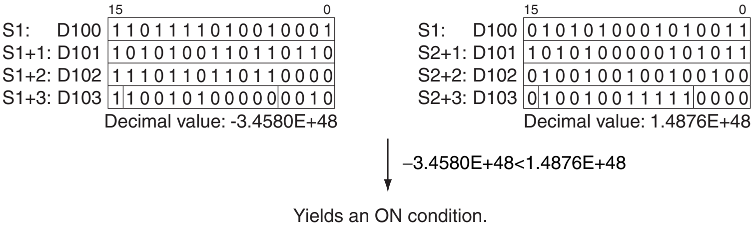


S1: D100 1 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0
S1+1: D101 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 0
S1+2: D102 1 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1
S1+3: D103 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1
Decimal value: 3.4580

S1: D100 0 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0
S2+1: D101 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0
S2+2: D102 1 1 0 0 1 1 0 1 0 0 1 1 0 1 0 1
S2+3: D103 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1
Decimal value: -1.4876

34580>14876

Does not yield an ON condition.



S1: D100 1 1 0 1 1 1 1 0 1 0 0 1 0 0 0 1
S1+1: D101 1 0 1 0 1 0 0 1 1 0 1 1 0 1 1 0
S1+2: D102 1 1 1 0 1 1 0 1 1 0 1 1 0 0 0 0
S1+3: D103 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0
Decimal value: -3.4580E+48

S1: D100 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1 1
S2+1: D101 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1
S2+2: D102 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
S2+3: D103 0 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0
Decimal value: 1.4876E+48

–3.4580E+48<1.4876E+48

Yields an ON condition.

Table Data Processing Instructions

Table Data Processing Instructions

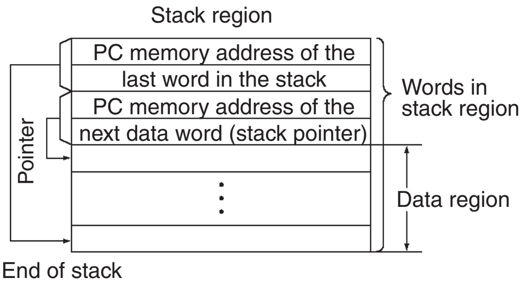
All of these instructions define or operate on a group of words. The group of words in a stack are defined by SSET(630), the group of words in a record-table are defined by DIM(631), and the group of words used in a range instruction are defined independently in each instruction.

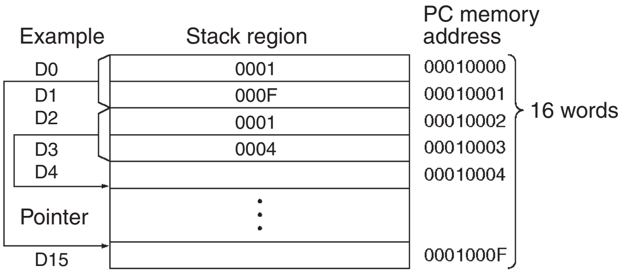
Group	Purpose	Instructions
Stack	Operate FIFO (first-in first-out) or LIFO (last-in first-out) data tables.	SSET(630), PUSH(632), FIFO(633), LIFO(634), SREAD(639), SWRIT(640), SINS(641), SDEL(642), and SNUM(638)
Record-table	Operate tables of data made up of records. (Record size is user-defined.)	DIM(631), SETR(635), and GETR(636)
Range	Operates on a range of words to find values such as the checksum, a particular value, the maximum value, or minimum value in the range.	FCS(180), SRCH(181), MAX(182), MIN(183), SUM(184), and SWAP(637)

● Stack Instructions

Stack instructions act on specially defined data tables called stacks. The first two words of the stack contain the PLC memory address of the last word in the stack and the second two words contain the stack pointer (the PLC memory address of the word that will be overwritten by the next PUSH(632) instruction).

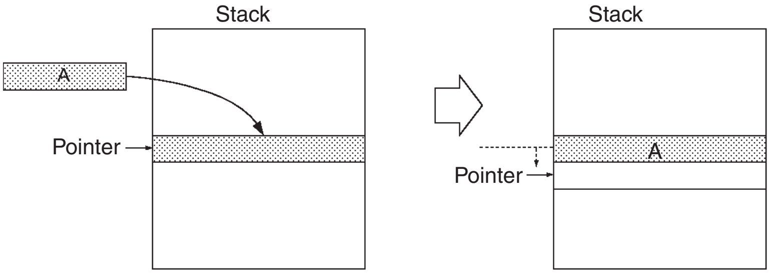
The following diagram shows the basic structure of a stack.





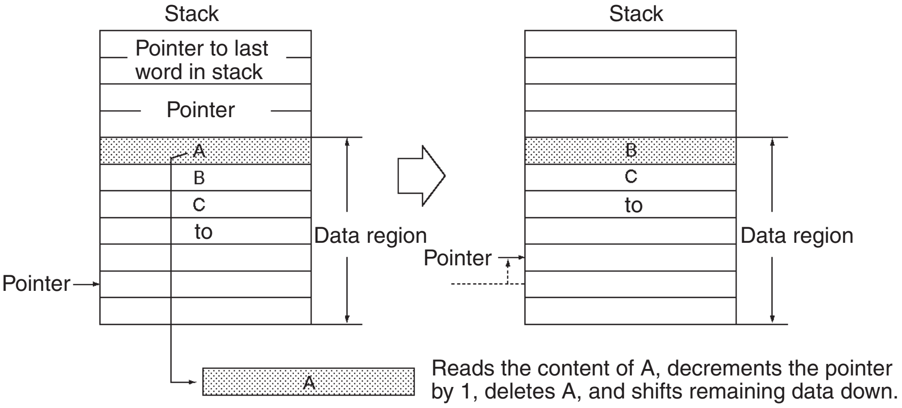
● PUSH(632)

Stores data in the address indicated by the stack pointer and increments the pointer by one.



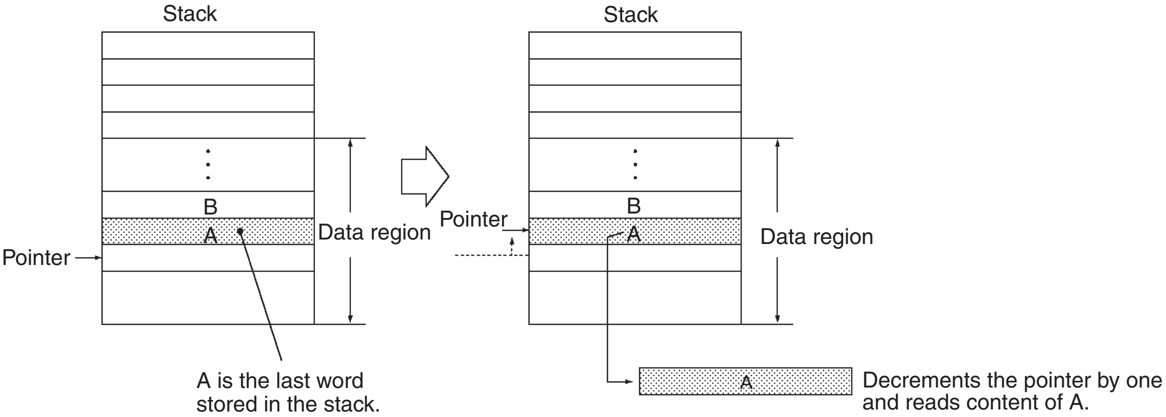
FIFO(633)

Reads first (oldest) word of data that was stored in the stack, shifts the remaining data down one word, and decrements the pointer by one.



LIFO(634)

Reads the last (most recent) word of data that was stored in the stack. Decrements the pointer by one and reads the data at this address (the most recent data stored in the stack). The read data will not be cleared.



To learn the number of data words in a stack, use an SNUM instruction.

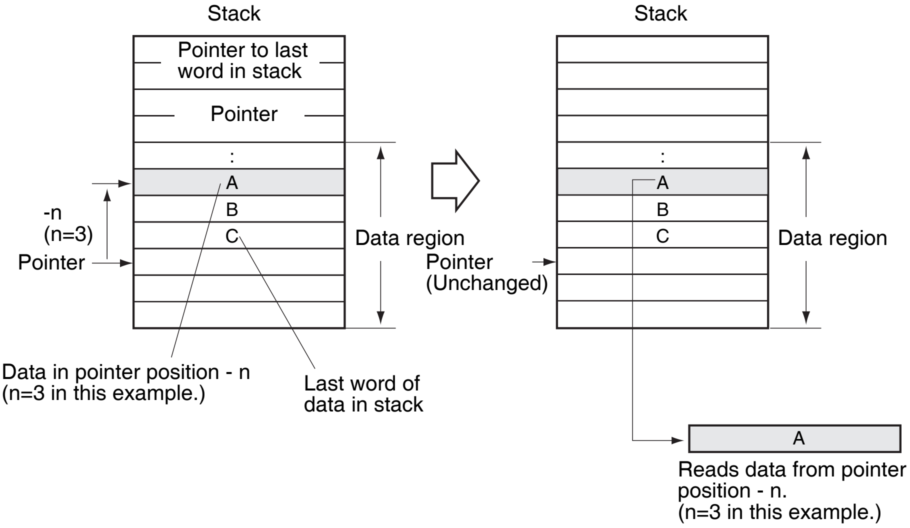
Example: Learning the number of items on a conveyor

To read (reference), change (refresh), insert, or delete a data word within the stack, use SREAD, SWRIT, SINS, and SDEL commands, respectively.

Example: To track items moving on a conveyor, treat the stack words as the items and manage additions, discharges, and changes of the items

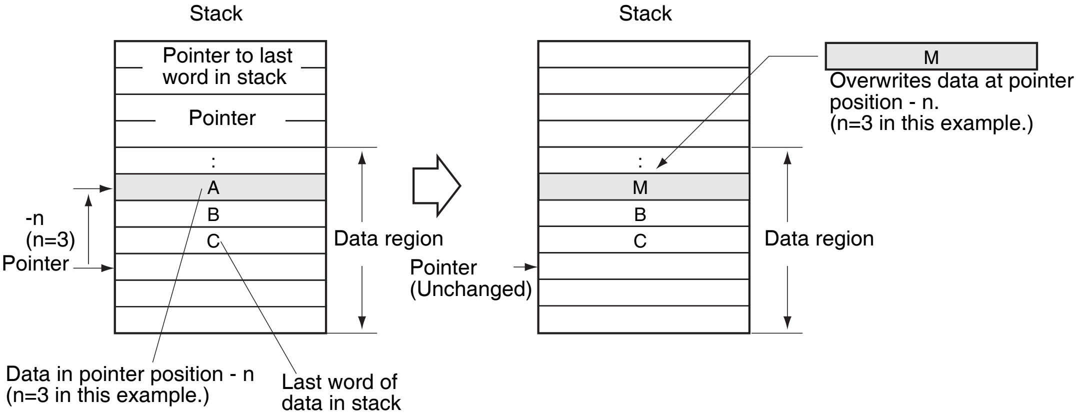
● SREAD(639)

Reads the data from the specified data element in the stack. The offset value indicates the location of the desired word (the number of words before the current pointer position).



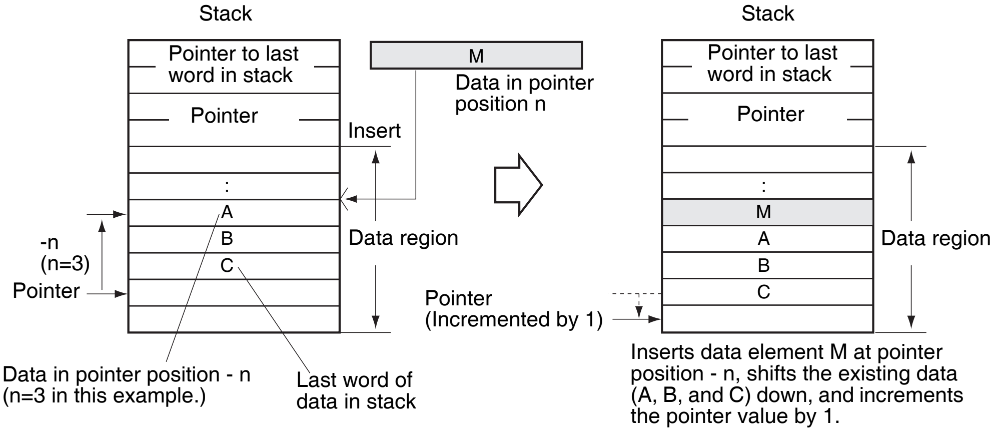
● SWRIT(640)

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 word (the number of words before the current pointer position).



● SINS(641)

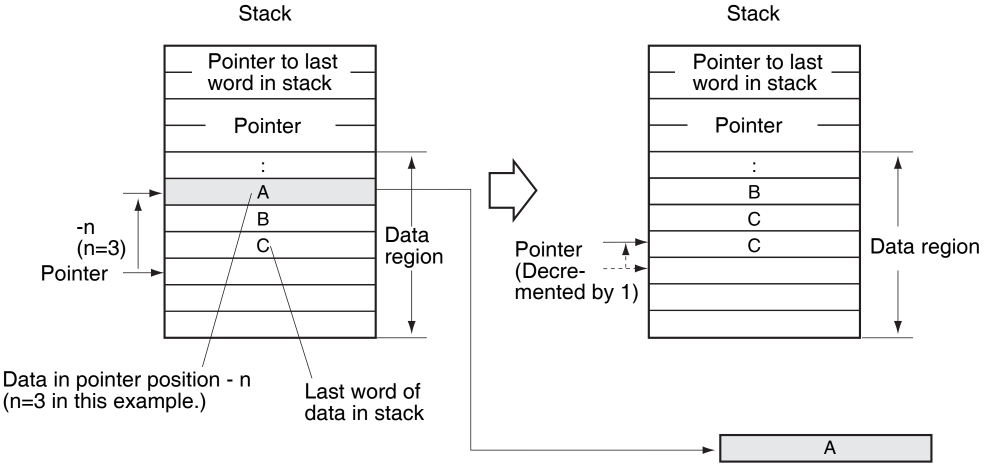
Inserts the source data at the specified location in the stack and shifts the rest of the data in the stack downward. The offset value indicates the location of the desired word (the number of words before the current pointer position).



Inserts data element M at pointer position - n, shifts the existing data (A, B, and C) down, and increments the pointer value by 1.

● SDEL(642)

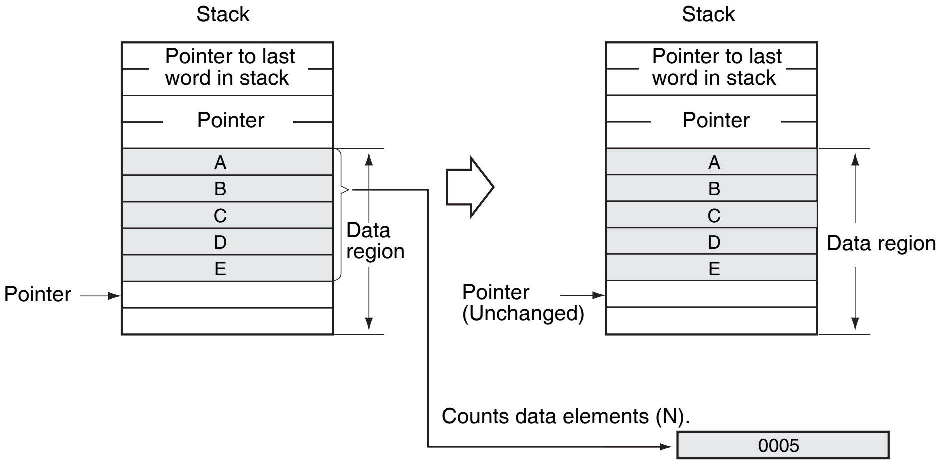
Deletes the data element at the specified location in the stack and shifts the rest of the data in the stack upward. The offset value indicates the location of the desired word (the number of words before the current pointer position).



Deletes the data element at pointer position - n, shifts the remaining data upward, and decrements the pointer value by 1.

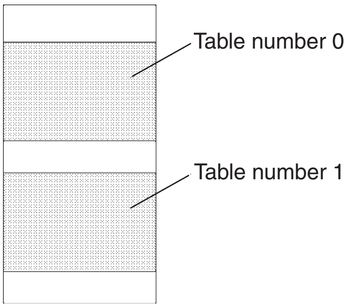
SNUM(638)

Counts the amount of stack data (number of words of data) from the stack pointer to the beginning of the data region.

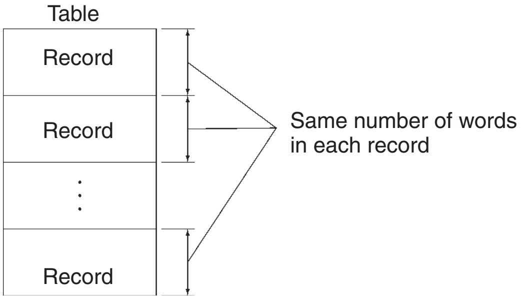


Record-table Instructions

A series of data consisting of more than one record with the same number of words in each record is called table data. Table data stored in the specified I/O memory are can be registered as the table area using the DIM instruction. Up to 16 separate tables can be defined with table numbers 0 to 15.



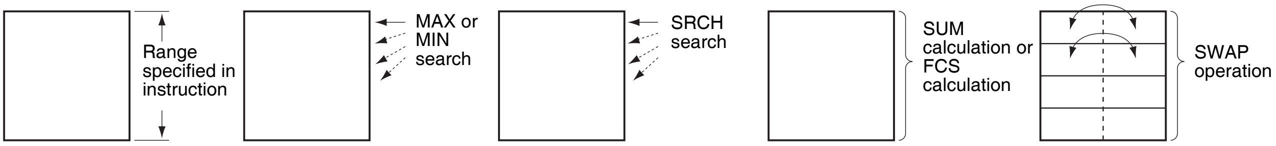
The following diagram shows the basic structure of a record table. Each record in a table has the same number of words.



Index Registers (IR) can be used to indirectly reference table data. Address calculation of the record can be easily made by using the SETR(635) (SET RECORD NUMBER) instruction and GETR(636) (GET RECORD NUMBER).

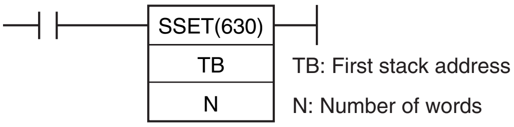
Range Instructions

The range instructions included here act on a specified range of words to find the maximum value (MAX(182)) or minimum value (MIN(183)), search for a particular value (SRCH(181)), calculate the sum (SUM(184)) or FCS (FCS(180)), or swap the contents of the leftmost and rightmost bytes in the words (SWAP(637)).



SSET

Instruction	Mnemonic	Variations	Function code	Function
SET STACK	SSET	@SSET	630	Defines a stack of the specified length beginning at the specified word.

Symbol	SSET
	

Applicable Program Areas

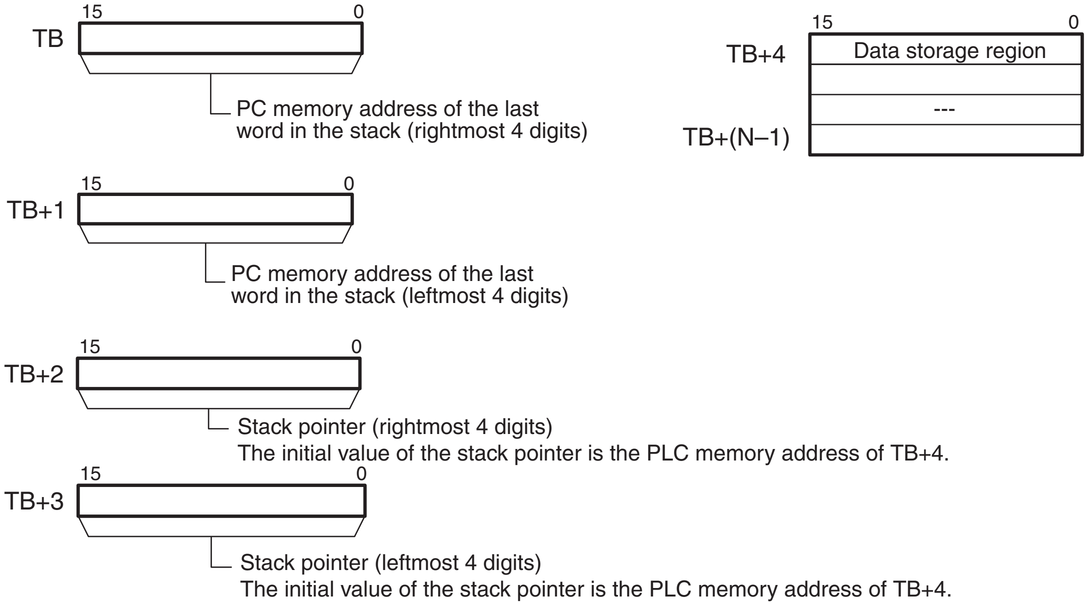
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
N	Number of words	UINT	1

TB: First stack address

TB through TB+3: Stack control words **TB+4 through TB+(N-1): Data storage region**



The initial value of the stack pointer is the PLC memory address of TB+4.

The initial value of the stack pointer is the PLC memory address of TB+4.

Note TB and TB+(N-1) must be in the same data area.

N: Number of words

This number specifies the stack area that contains the stack control words and the data storage region. N must be between 5 and 65535.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
N											OK	OK						

Flags

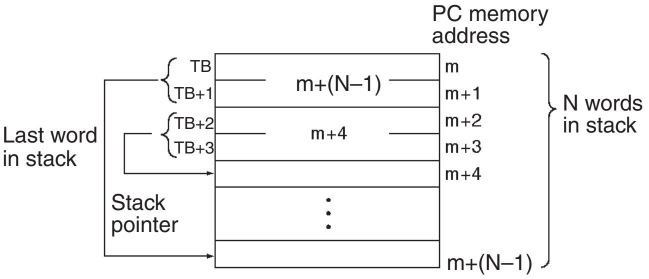
Name	Label	Operation
Error Flag	ER	• ON if N is not within the specified range of 0005 to FFFF. • OFF in all other cases.

Note 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.

Function

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.



Hint

SSET(630) just establishes and initializes a stack. Use the following instructions to store in the stack and read data from the stack.

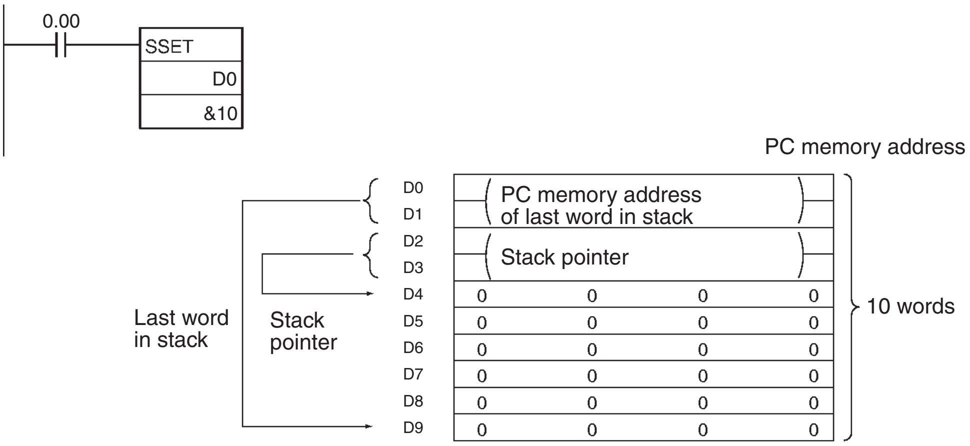
1. PUSH(632) stores data in the stack one word at a time.
2. FIFO(633) and LIFO(634) read data from the stack. FIFO(633) reads the first word that was stored; LIFO(634) reads the last word that was stored.
3. The stack pointer value in the stack control word is automatically updated when PUSH(632), FIFO(633), 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.

Related instructions

- PUSH (stack data store) instruction: Stores data in the specified stack area.
- FIFO (first in, first out) instruction: Reads the first data word that was stored in the specified stack area.
- LIFO (last in, first out): Reads the last data word that was stored in the stack.

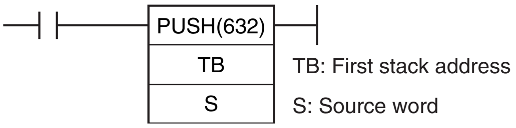
Example Programming

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.



PUSH

Instruction	Mnemonic	Variations	Function code	Function
PUSH ONTO STACK	PUSH	@PUSH	632	Writes one word of data to the specified stack.

	PUSH
Symbol	

Applicable Program Areas

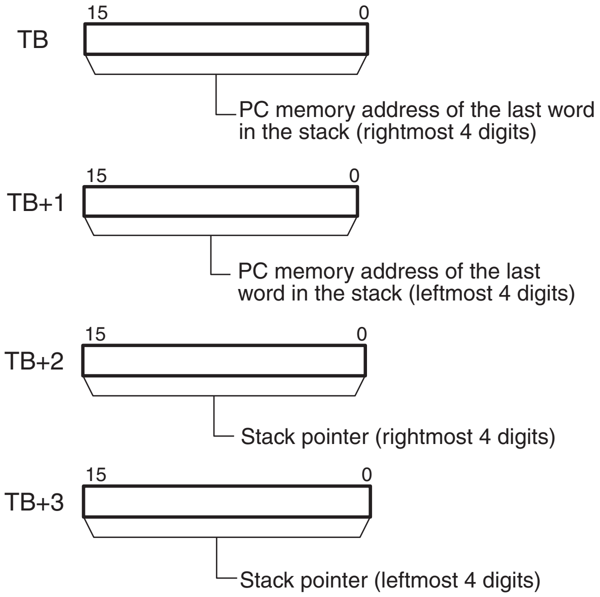
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

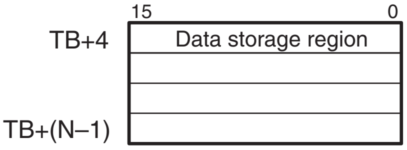
Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
S	Source word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



TB+4 through TB+(N-1): Data storage region



● Operand Specifications

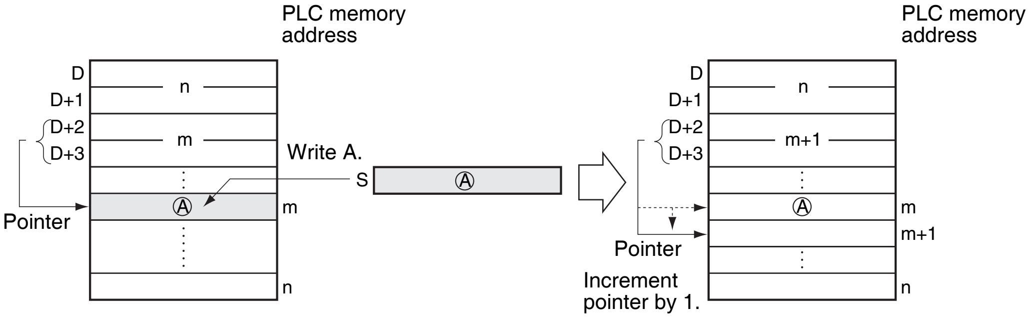
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S											OK	OK						

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.

Function

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.
The stack must be defined in advance with SSET(630).

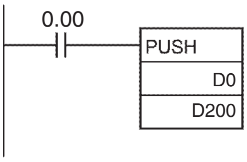


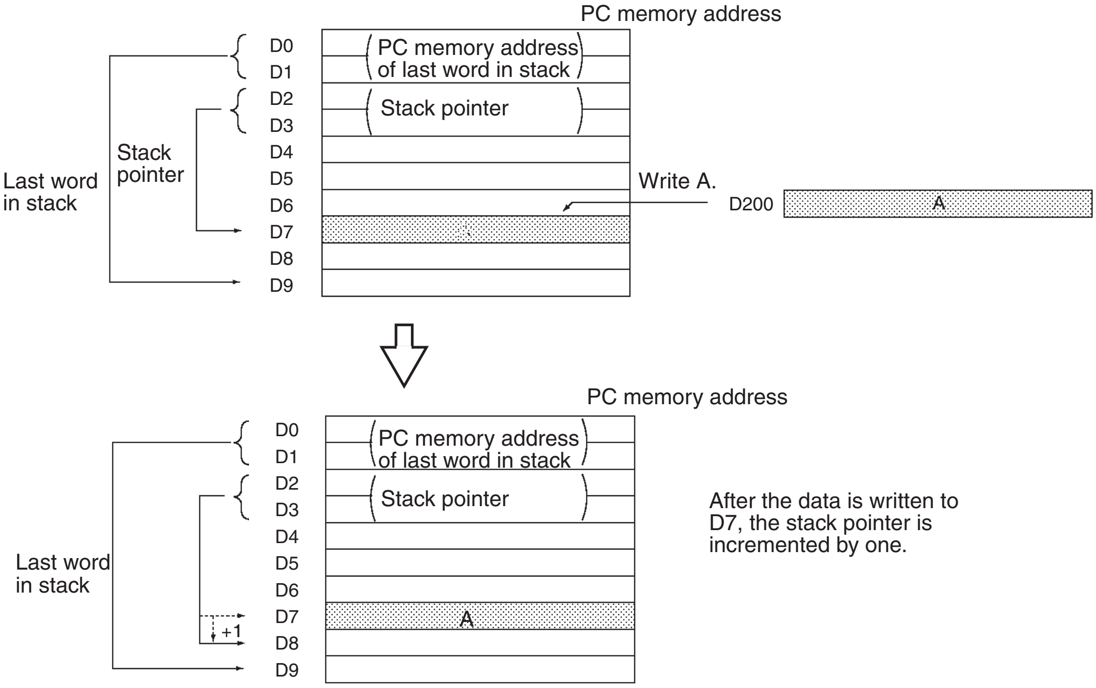
Hint

- After data is stored with this instruction, the stack pointer specifies the next address after the last data word.
- 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.

Example Programming

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.

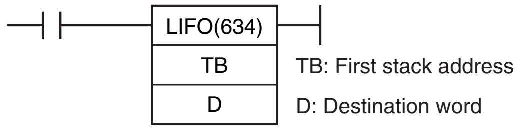
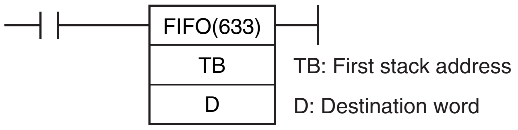




After the data is written to D7, the stack pointer is incremented by one.

LIFO/FIFO

Instruction	Mnemonic	Variations	Function code	Function
LAST IN FIRST OUT	LIFO	@LIFO	634	Reads the last word of data written to the specified stack (the newest data in the stack).
FIRST IN FIRST OUT	FIFO	@FIFO	633	Reads the first word of data written to the specified stack (the oldest data in the stack).

	LIFO	FIFO
Symbol	 TB: First stack address D: Destination word	 TB: First stack address D: Destination word

Applicable Program Areas

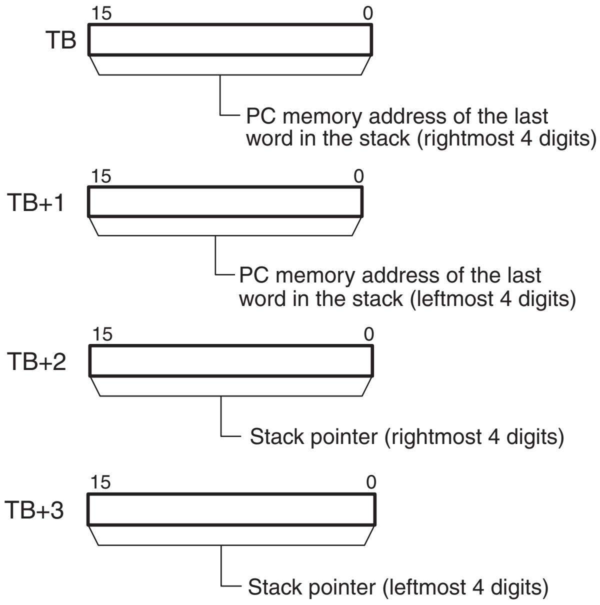
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

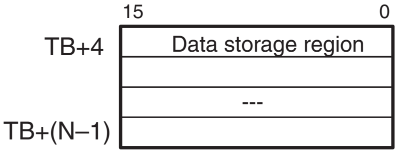
Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
D	Destination word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



TB+4 through TB+(N-1): Data storage region



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D												OK						

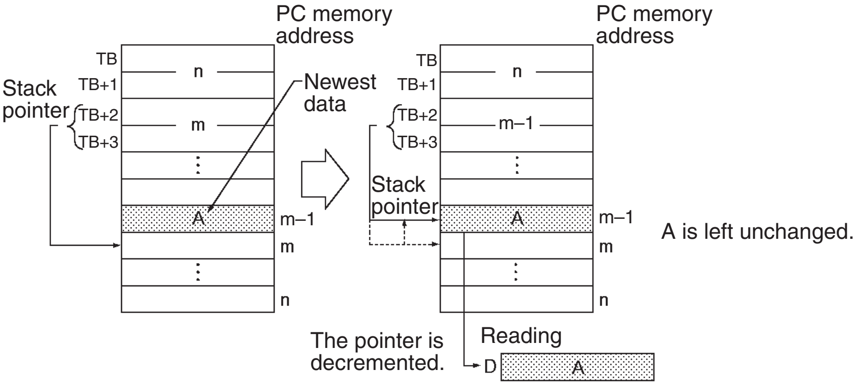
Flags

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.

Function

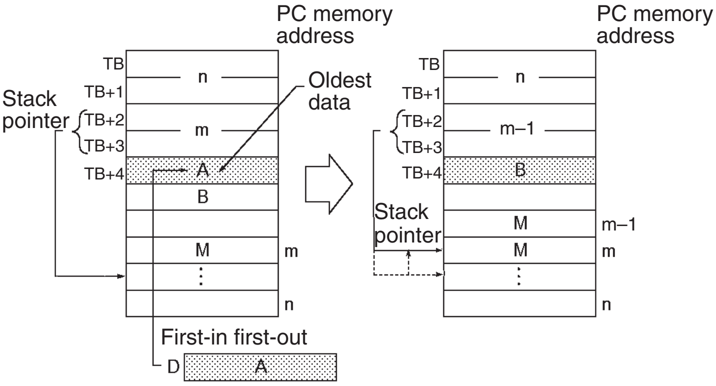
● LIFO

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.
The stack must be defined in advance with SSET(630).



● FIFO

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.
The stack must be defined in advance with SSET(630).



Hint

● LIFO

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.

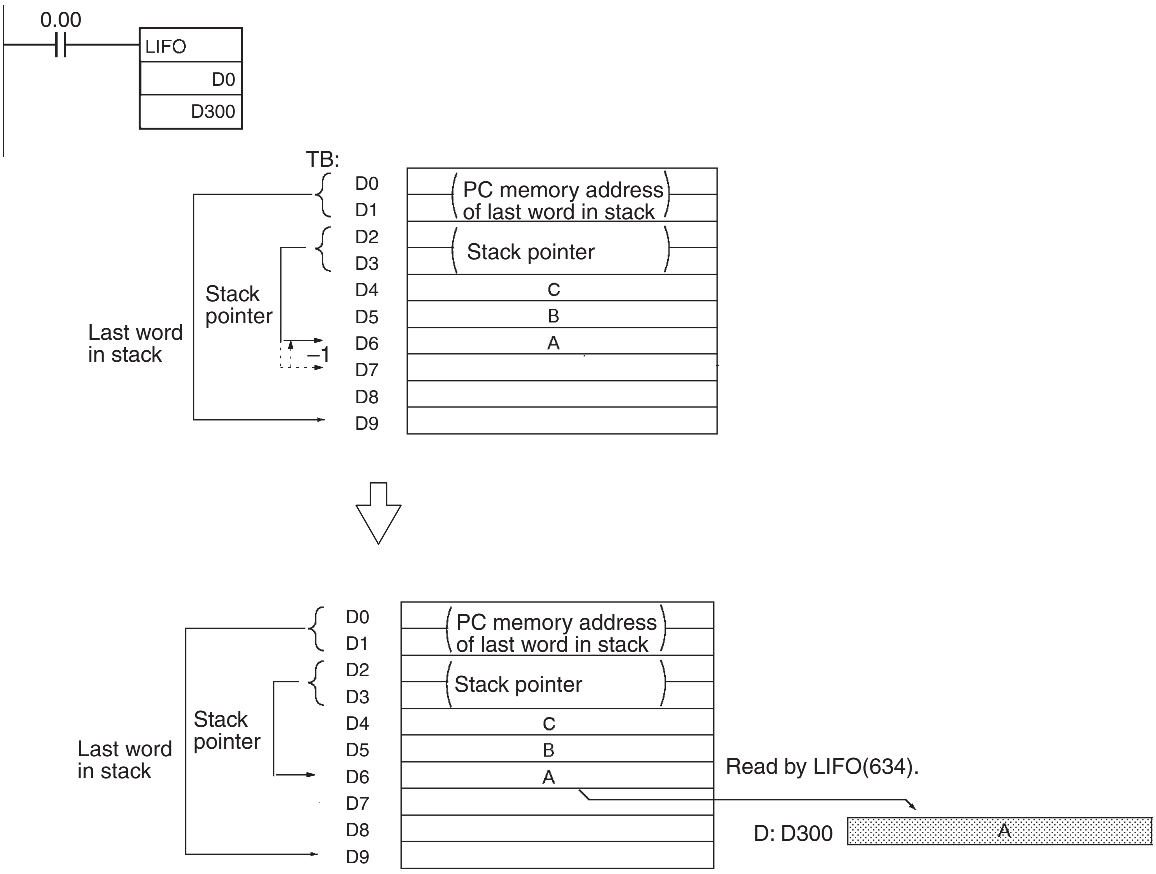
● FIFO

- 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.

Example Programming

● LIFO

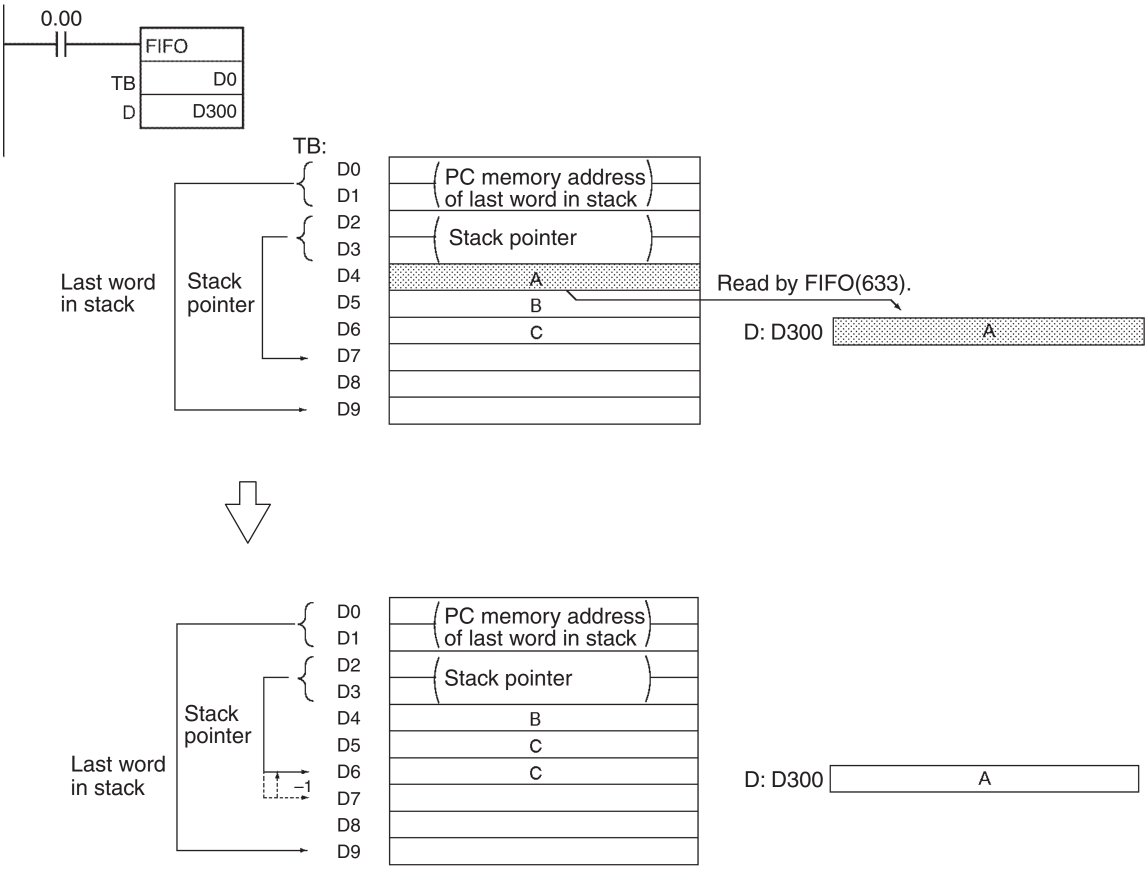
When CIO 0.00 is ON in the following example, LIFO(634) reads the content of the word indicated by the stack pointer (D6) and writes that data to D300.



After the data is written to D300, the stack pointer is decremented by one. The content of D6 is left unchanged.

- **FIFO**

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.



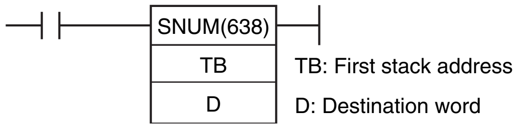
After the data is written to D300, the stack pointer is decremented by one and the remaining data is shifted down. (The content of D5 is shifted to D4 and the content of D6 is shifted to D5.)

SNUM

Instruction	Mnemonic	Variations	Function code	Function
STACK SIZE READ	SNUM	@SNUM	638	Counts the amount of stack data (number of words) in the specified stack.

Symbol

SNUM



Applicable Program Areas

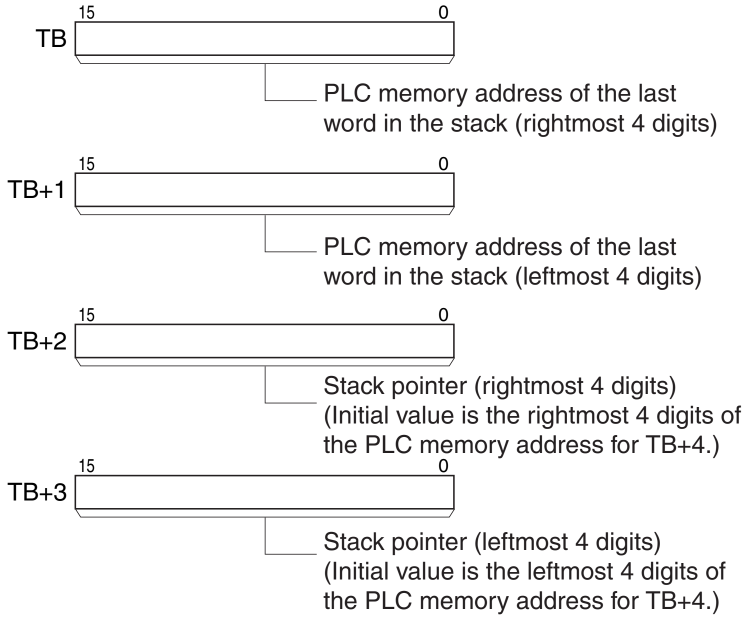
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

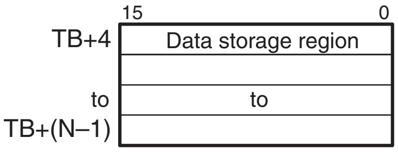
Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
D	Destination word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



TB+4 through TB+(N-1): Data storage region



● Operand Specifications

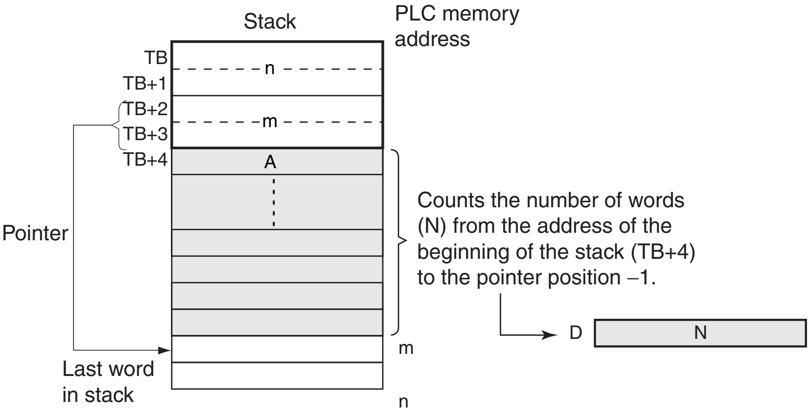
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	OFF
Equal Flag	P_EQ	• ON if the number of words of data in the stack (the value output to D) is 0. • OFF in all other cases.

Function

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.
The stack must be defined in advance with SSET(630).

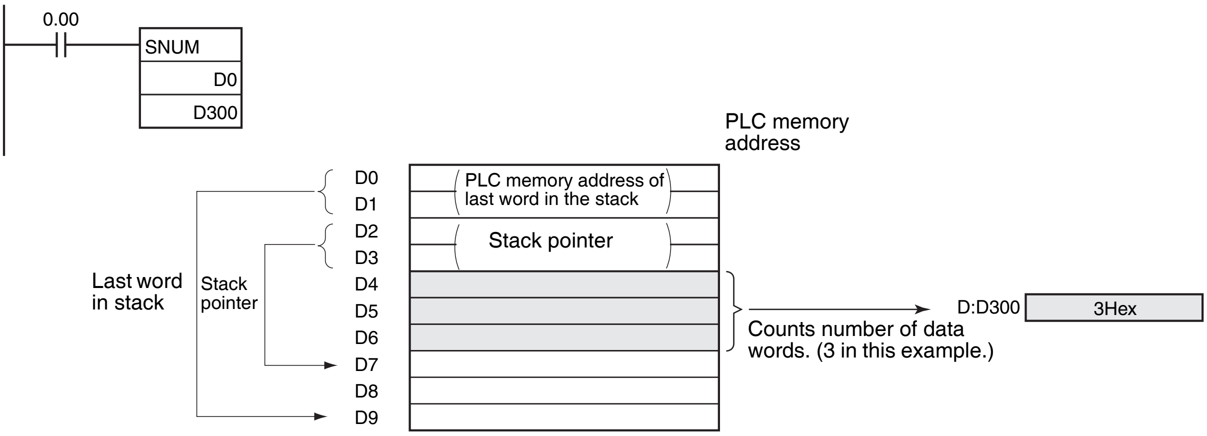


Hint

As an example, this instruction is used to count the number of work pieces currently on a conveyor belt.

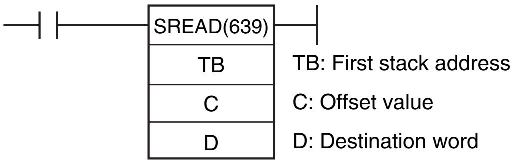
Example Programming

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.)



SREAD

Instruction	Mnemonic	Variations	Function code	Function
STACK DATA READ	SREAD	@SREAD	639	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).

	SREAD
Symbol	

TB: First stack address
C: Offset value
D: Destination word

Applicable Program Areas

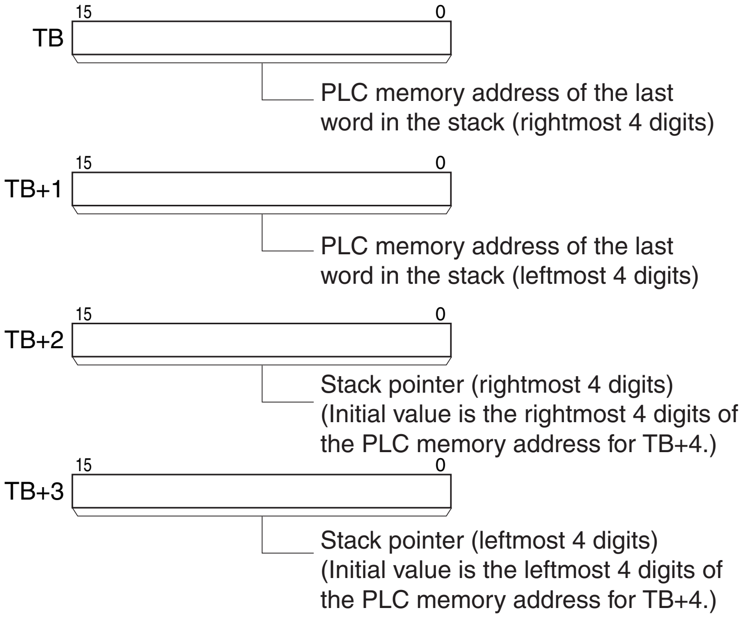
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
C	Offset value	UINT	1
D	Destination word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



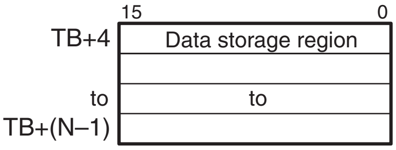
PLC memory address of the last word in the stack (rightmost 4 digits)

PLC memory address of the last word in the stack (leftmost 4 digits)

Stack pointer (rightmost 4 digits) (Initial value is the rightmost 4 digits of the PLC memory address for TB+4.)

Stack pointer (leftmost 4 digits) (Initial value is the leftmost 4 digits of the PLC memory address for TB+4.)

TB+4 through TB+(N-1): Data storage region



● Operand Specifications

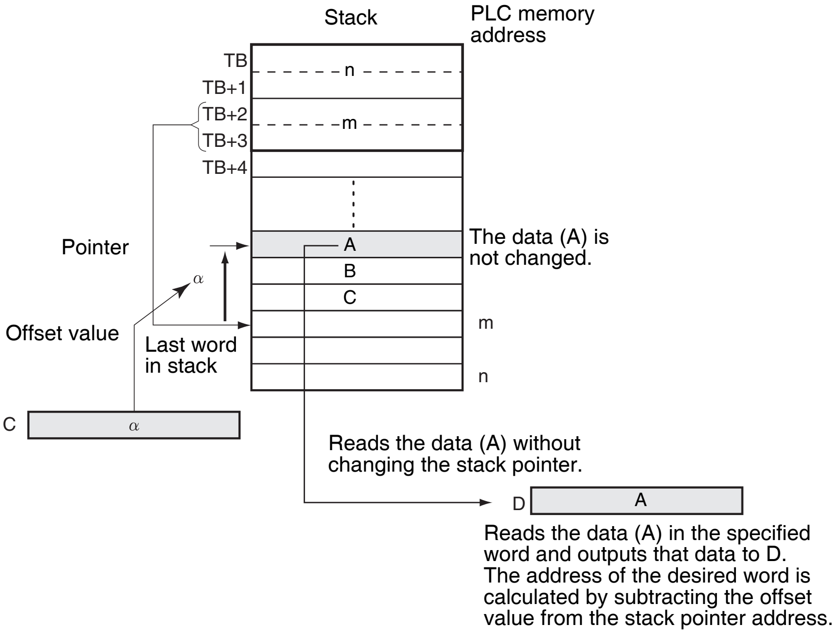
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK	OK						
D											---							

Flags

Name	Label	Operation
Error Flag	P_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). • 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.
Equal Flag	P_EQ	• ON if the output data in D is 0. • OFF in all other cases.

Function

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.
The stack must be defined in advance with SSET(630).

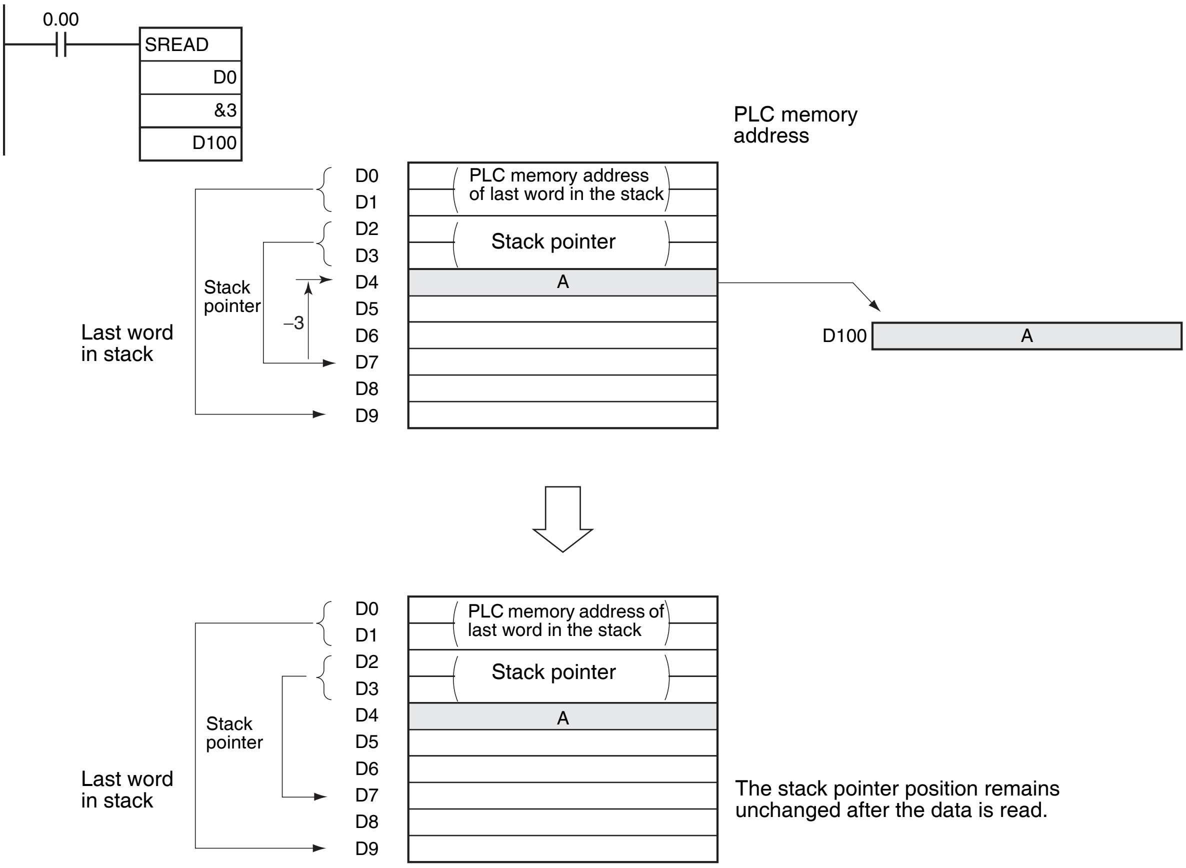


Hint

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.

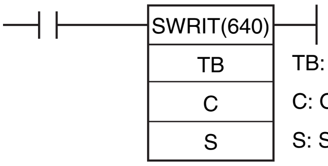
Example Programming

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.



SWRIT

Instruction	Mnemonic	Variations	Function code	Function
STACK DATA OVERWRITE	SWRIT	@SWRIT	640	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).

	SWRIT
Symbol	 TB: First stack address C: Offset value S: Source word

Applicable Program Areas

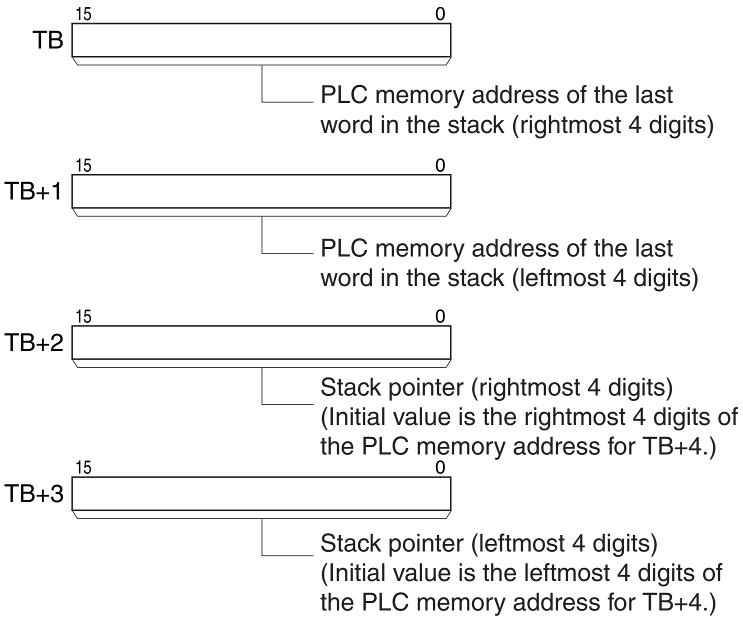
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

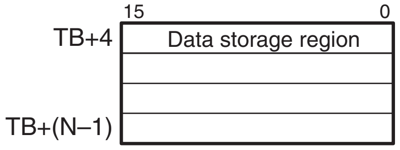
Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
C	Offset value	UINT	1
S	Source word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



TB+4 through TB+(N-1): Data storage region



● Operand Specifications

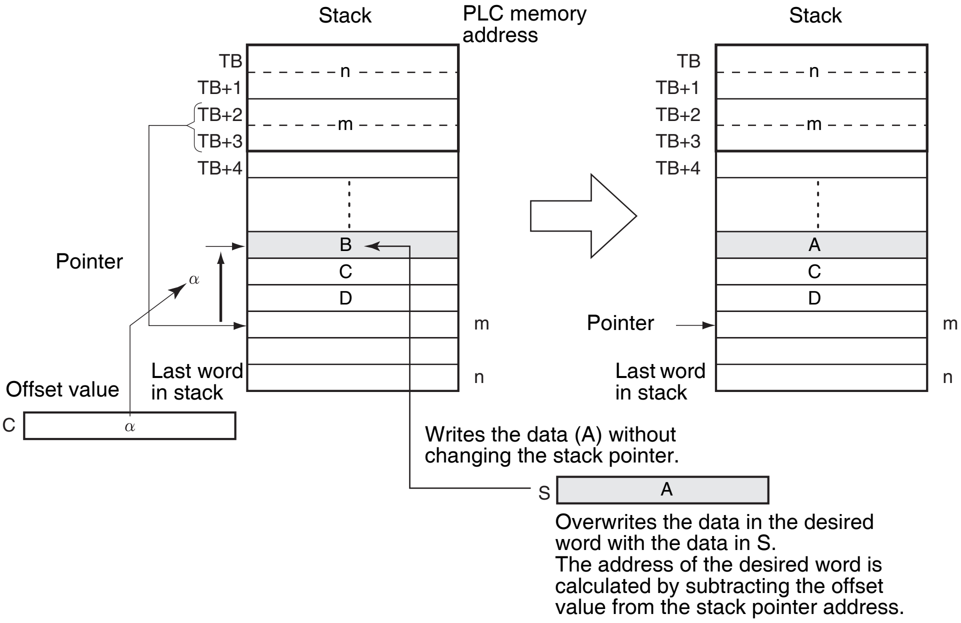
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_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). • 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 under flow error.) • OFF in all other cases.

Function

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. The stack must be defined in advance with SSET(630).



Writes the data (A) without changing the stack pointer.

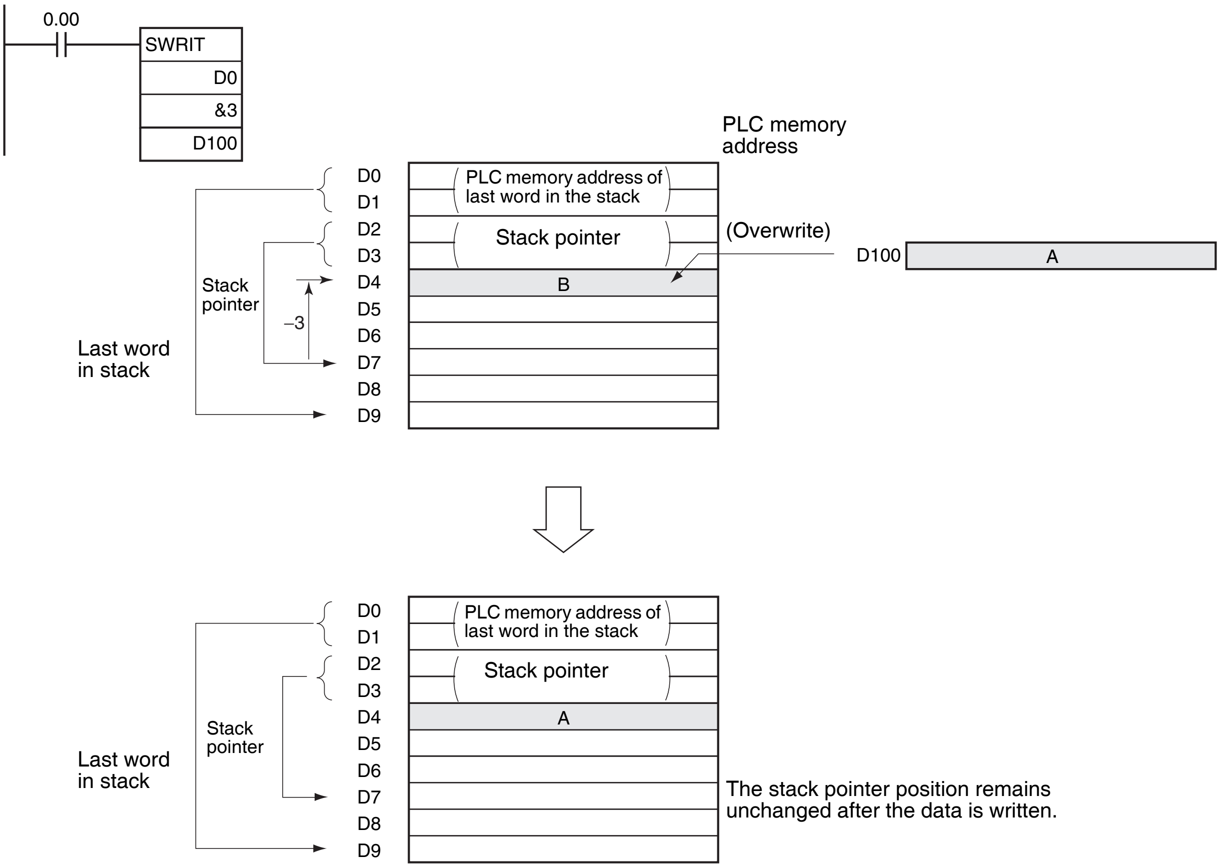
Overwrites the data in the desired word with the data in S.
The address of the desired word is calculated by subtracting the offset value from the stack pointer address.

Hint

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.

Example Programming

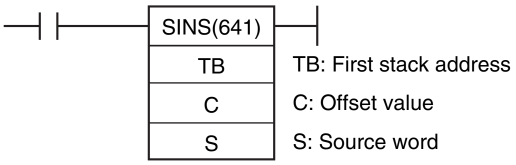
When CIO 0.00 is ON in the following example, SWRIT(640) writes the data in D100 to the specified word in the stack starting at D0. In this case, the stack pointer indicates D7 and the offset value is 3, so the data in D4 is overwritten. SWRIT(640) does not change the pointer.



SINS

Instruction	Mnemonic	Variations	Function code	Function
STACK DATA INSERT	SINS	@SINS	641	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).

Symbol — **SINS**



Applicable Program Areas

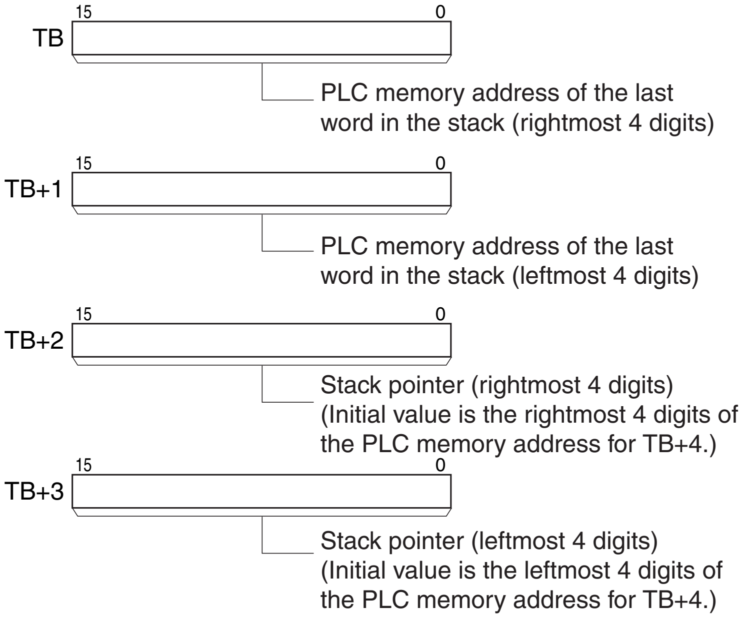
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

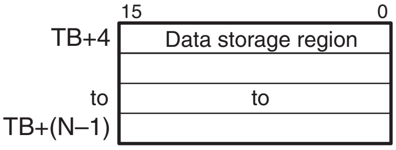
Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
C	Offset value	UINT	1
S	Source word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



TB+4 through TB+(N-1): Data storage region



Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK	OK						
S																		

Flags

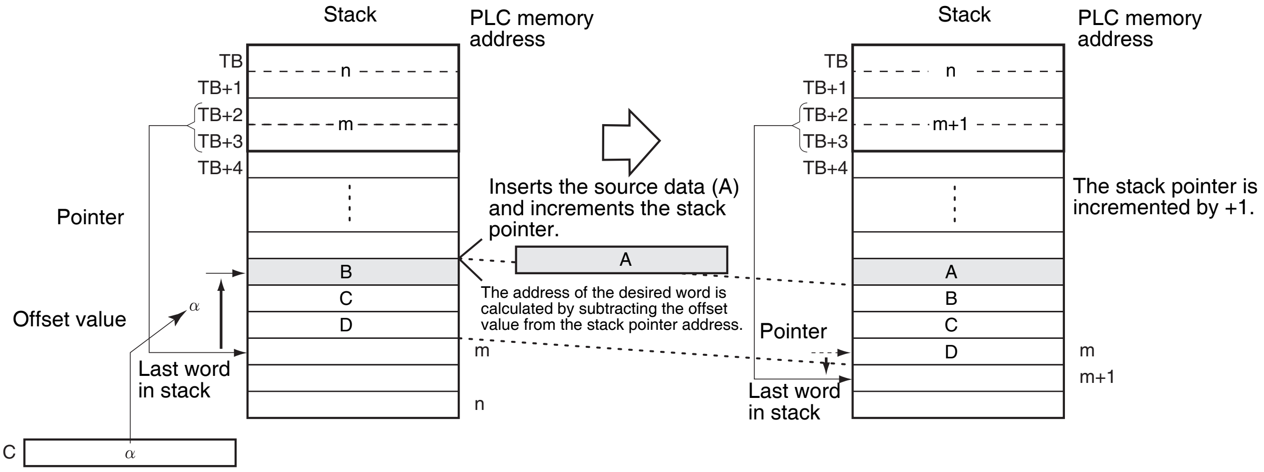
Name	Label	Operation
Error Flag	P_ER	• ON if the address indicated by the stack pointer (TB+3 and TB+2) is greater than the PLC memory address of last word in the data region of the stack. (This is a stack overflow error.) • ON if the offset value specified is greater than the maximum data region size - 1 (FFFA hex). • OFF in all other cases.

Function

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) 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.

The stack must be defined in advance with SSET(630).

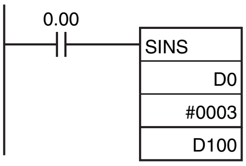


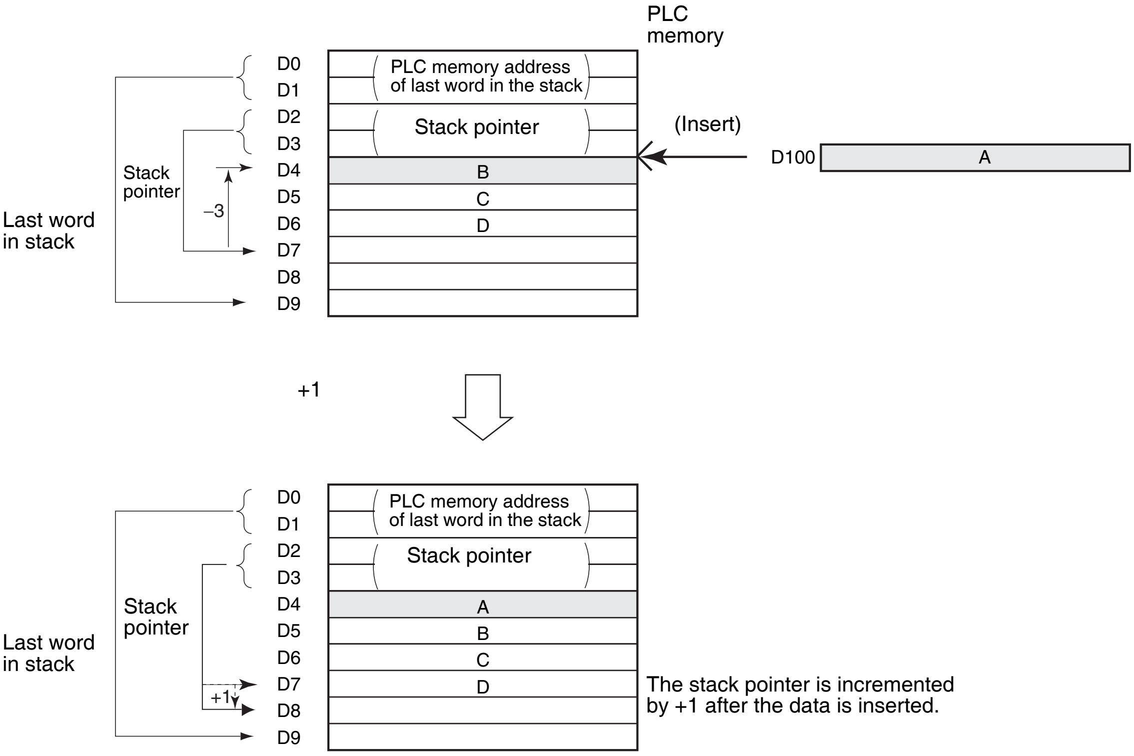
Hint

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.

Example Programming

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.



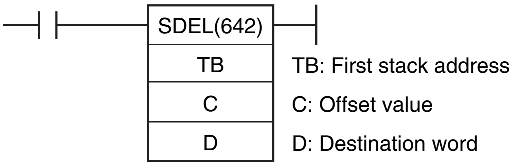


SDEL

Instruction	Mnemonic	Variations	Function code	Function
STACK DATA DELETE	SDEL	@SDEL	642	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).

Symbol

SDEL



Applicable Program Areas

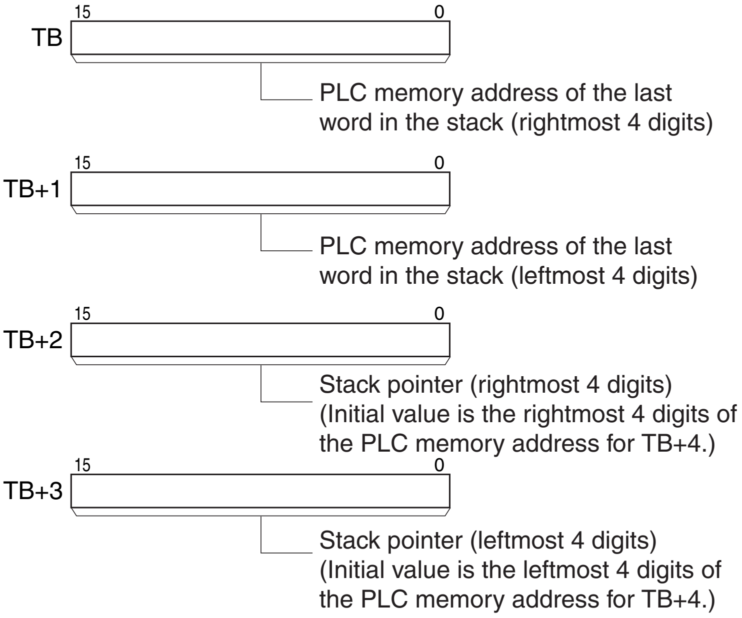
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

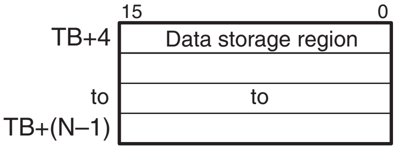
Operand	Description	Data type	Size
TB	First stack address	UINT	Variable
C	Offset value	UINT	1
D	Destination word	UINT	1

TB: First stack address

TB through TB+3: Stack control words



TB+4 through TB+(N-1): Data storage region



Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK	OK						
D											---							

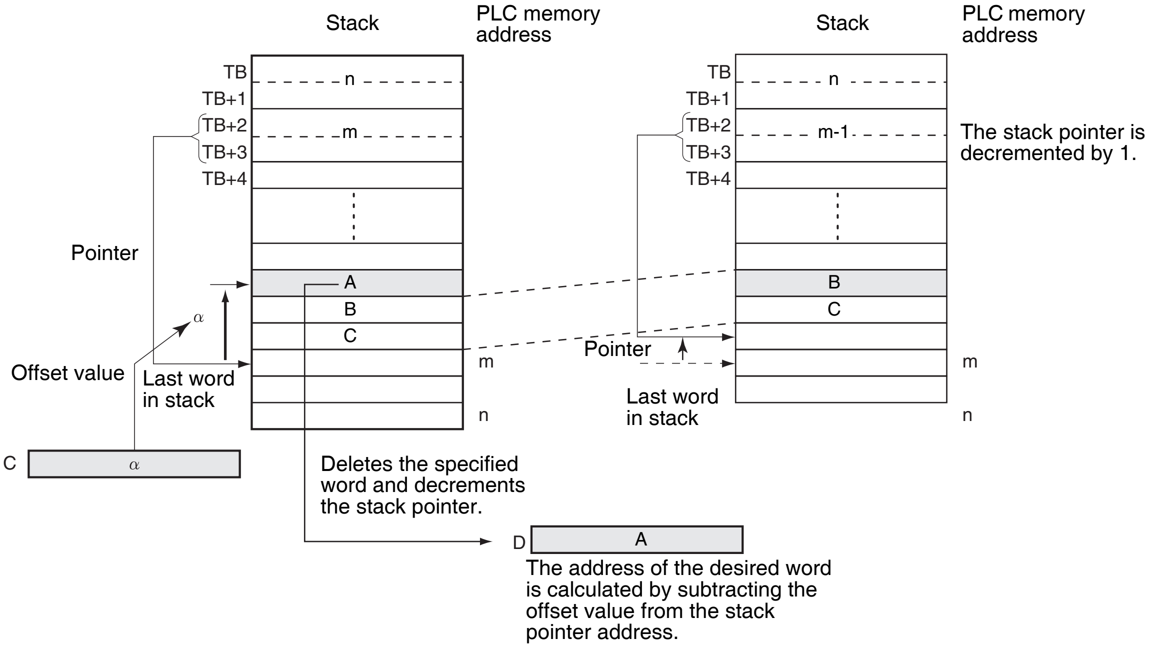
Flags

Name	Label	Operation
Error Flag	P_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.
Equal Flag	P_EQ	• ON if the output data in D is 0000. • OFF in all other cases.

Function

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.

The stack must be defined in advance with SSET(630).

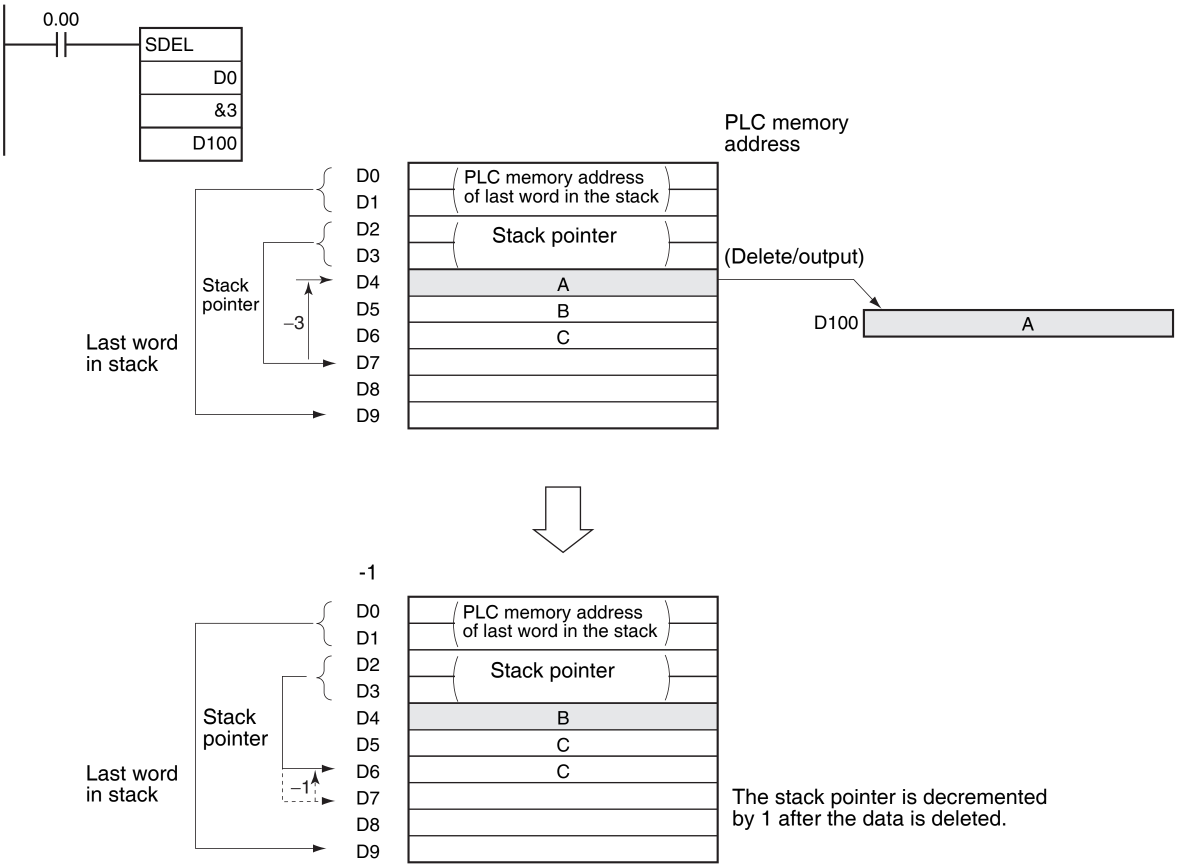


Hint

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.

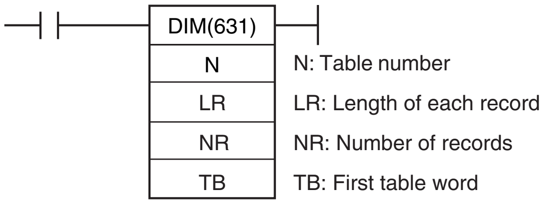
Example Programming

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.



DIM

Instruction	Mnemonic	Variations	Function code	Function
DIMENSION RECORD TABLE	DIM	@DIM	631	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.

Symbol	DIM
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Table number	---	1
LR	Length of each record	UINT	1
NR	Number of records	UINT	1
TB	First table word	UINT	Variable

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).

● Operand Specifications

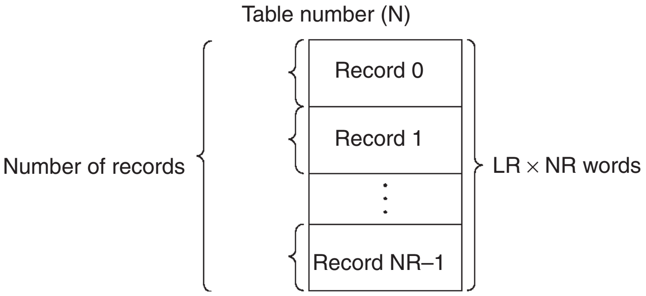
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
LR, NR	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK		OK				
D											---	---						

Flags

Name	Label	Operation
Error Flag	ER	• ON if LR or NR is 0. • OFF in all other cases.

Function

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.
Records in a registered table are identified by their record numbers, which range from 0 to NR-1.



Hint

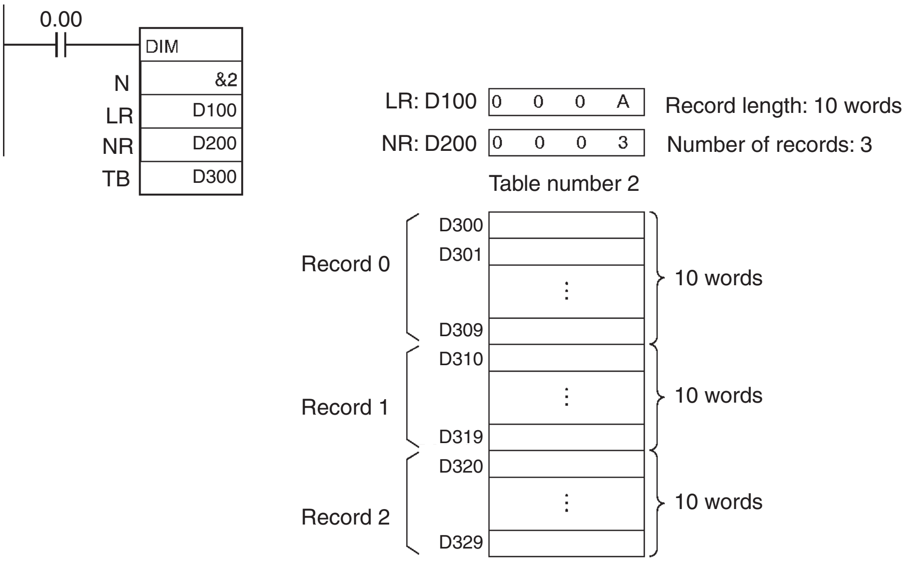
- Use DIM(631) in combination with SETR(635) (SET RECORD NUMBER) or GETR(636) (GET RECORD NUMBER) to simplify the calculation of addresses in data tables. Use DIM(631) to divide data into records and then use SETR(635) to store the first address of the desired record in an Index Register. The Index Register 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.

Related instructions

- 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).

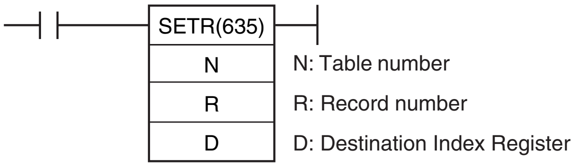
Example Programming

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.



SETR

Instruction	Mnemonic	Variations	Function code	Function
SET RECORD LOCATION	SETR	@SETR	635	Writes the location of the specified record (the PLC memory address of the beginning of the record) in the specified Index Register.

	SETR
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Table number	---	1
R	Record number	UINT	1
D	Destination Index Register	WORD	2

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

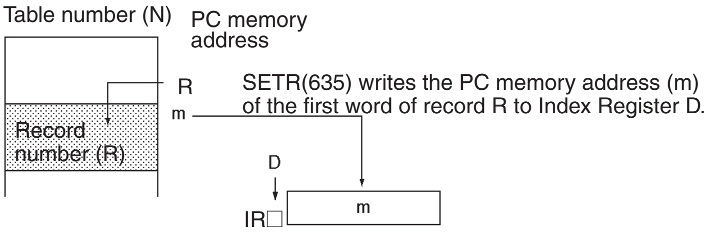
Area	Word addresses								Indirect DM/EM addresses		Con- stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
R	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK		OK				
D	---	---	---	---	---	---	---	---	---	---	---	---	OK	---				

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.

Function

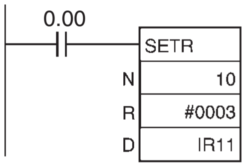
SETR(635) stores the PLC memory address of the first word of the specified record in the specified Index Register.

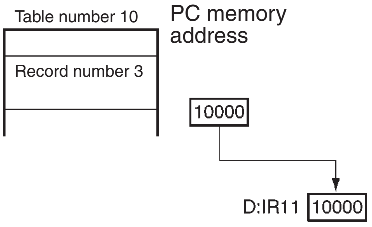


Note
- The record table must be defined in advance with DIM(631).
- Valid record numbers range from 0 to NR-1, where NR is the number of records specified when the table was defined with DIM(631).

Example Programming

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.



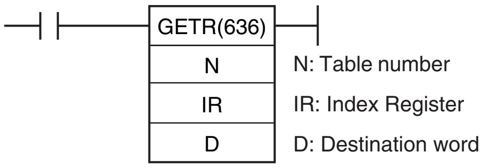


GETR

Instruction	Mnemonic	Variations	Function code	Function
GET RECORD NUMBER	GETR	@GETR	636	Returns the record number of the record at the PLC memory address contained in the specified Index Register.

Symbol

GETR



Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Table number	---	1
IR	Index Register	UINT	2
D	Destination word	WORD	1

N: Table number

Indicates the table number. N must be between 0 and 15.

IR: Index Register

Indicates the desired Index Register. IR must be IR0 to IR15.

● Operand Specifications

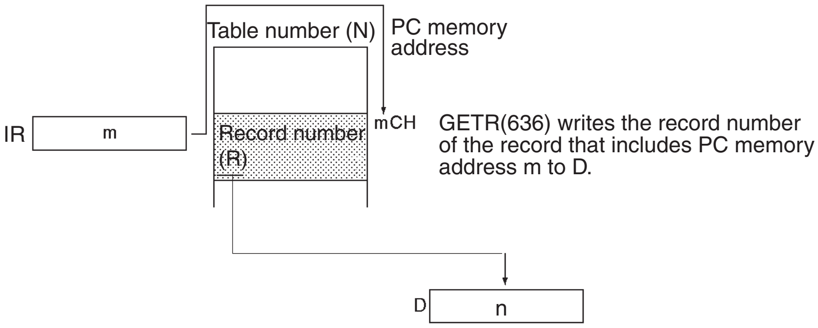
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
IR											---		OK					
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK	---	OK				

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.

Function

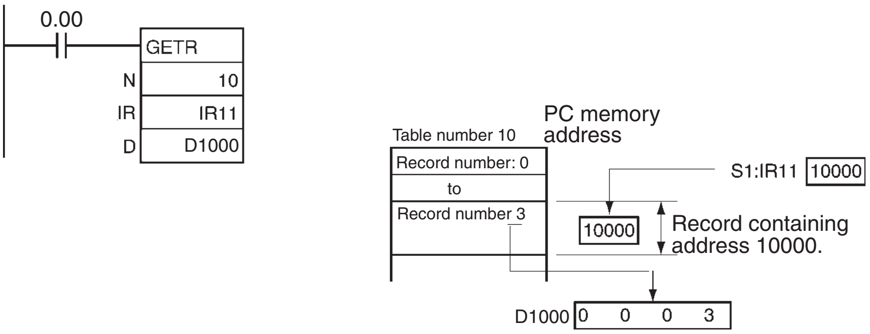
GETR(636) finds which record includes the PLC memory address contained in the specified Index Register and writes that record number in D. The PLC memory address contained in the Index Register does not have to be the first word in the record; it can be any word in the record.



Note The record table must be defined in advance with DIM(631).

Example Programming

When CIO 0.00 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.

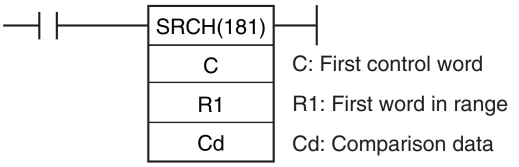


SRCH

Instruction	Mnemonic	Variations	Function code	Function
DATA SEARCH	SRCH	@SRCH	181	Searches for a word of data within a range of words.

Symbol

SRCH



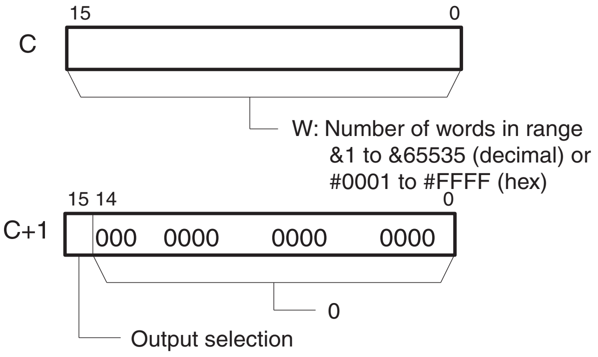
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

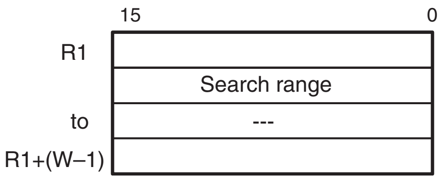
Operand	Description	Data type	Size
C	First control word	UDINT	2
R1	First word in range	UINT	Variable
Cd	Comparison data	WORD	1

C: First control word



0: Does not output number of matches to DR0.
1: Outputs number of matches to DR0.

R1: First word in range



Note C and C+1, R1 and R1+W-1 must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
Cd											OK	OK						

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.
Equal Flag	=	• ON if one or more of the words in the search range contain the comparison data. • OFF in all other cases.

Related Auxiliary Area Words and Bits

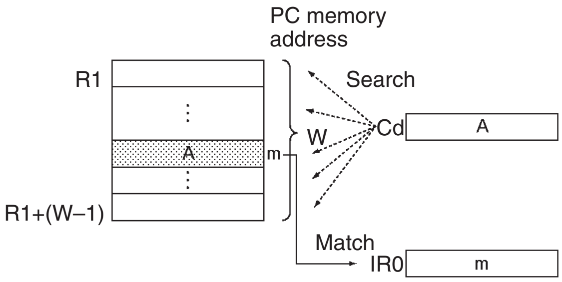
Name	Address	Operation
IR0 Output for Background Execution	A595 and A596	When an index register is specified as the output for an instruction processed in the background, A595 and A596 receive the output instead of IR0. (A595 contains the rightmost digits, and A596 contains the leftmost digits.)
DR0 Output for Background Execution	A597	When a data register is specified as the output for an instruction processed in the background, A597 receives the output instead of DR0.
Equals Flag for Background Execution	A598.01	This flag is turned ON if matching data is found for a SRCH(181) instruction executed in the background.
ER/AER Flag for Background Execution	A395.10	This flag is turned ON if an error or illegal access occurs during background execution.

Function

SRCH(181) searches the range of memory from R1 to R1+W-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.



Hint

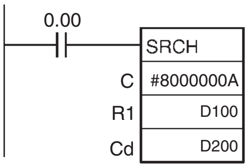
- SRCH(181) searches table data that contains one word in each record. For searching data that contains more than one word per record, use DIM(631), SETR(635), GETR(636), FOR(512)-NEXT(513), or BREAK(514) together with an Index Register (IR).
- SRCH(181) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

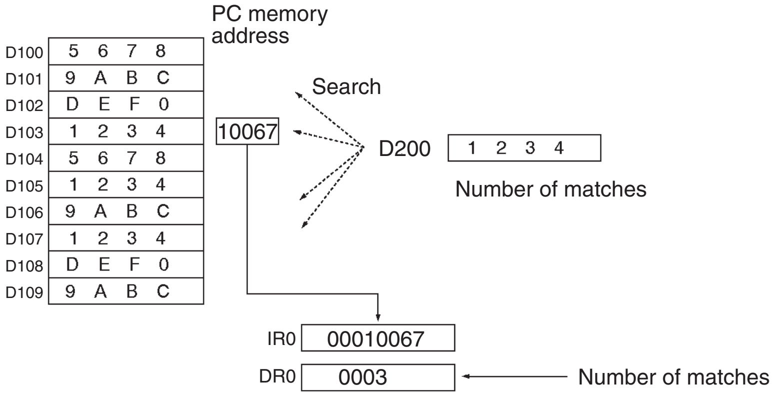
Precautions

- The status of the Equals Flag can be checked immediately after execution to determine whether or not there was a match.
- If no match is found, the contents of IR0 and DR0 are left unchanged.
- If background execution is enabled in the PLC Setup, the PLC memory address of the first word containing a match will be output to Auxiliary Area words A595 and A596 instead of IR0.
- If background execution is enabled in the PLC Setup and control word C+1 is set to output the total number of matches to DR0 (C+1 = 8000 hex), the total number of matches will be output to Auxiliary Area word A597 instead of DR0.

Example Programming

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.



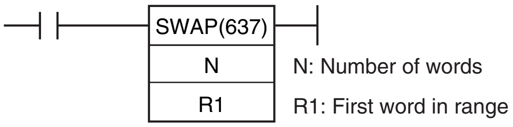


D100	5 6 7 8
D101	9 A B C
D102	D E F 0
D103	1 2 3 4
D104	5 6 7 8
D105	1 2 3 4
D106	9 A B C
D107	1 2 3 4
D108	D E F 0
D109	9 A B C

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.

SWAP

Instruction	Mnemonic	Variations	Function code	Function
SWAP BYTES	SWAP	@SWAP	637	Switches the leftmost and rightmost bytes in all of the words in the range.

Symbol	SWAP
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

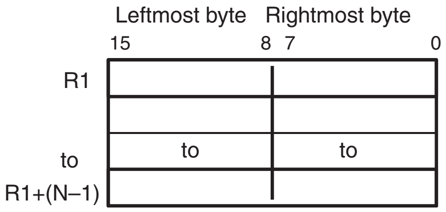
Operands

Operand	Description	Data type	Size
N	Number of words	UINT	1
R1	First word in range	UINT	Variable

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



Note R1 and R1+(N-1) must be in the same data area.

● Operand Specifications

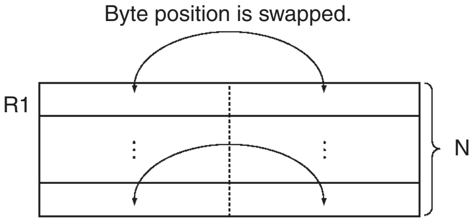
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
R1											---	---						

Flags

Name	Label	Operation
Error Flag	ER	• ON if the N is 0. • 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.

Function

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.

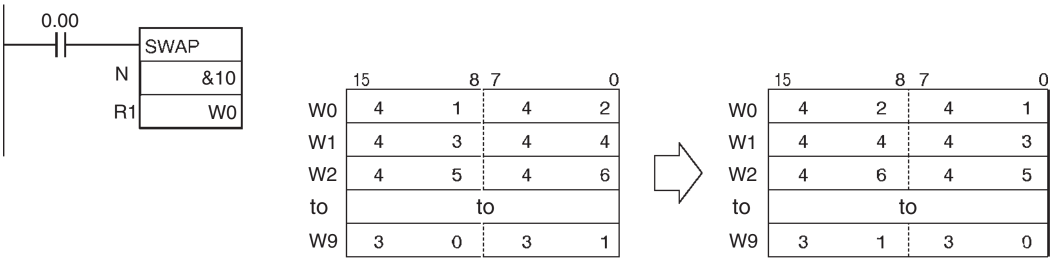


Hint

- This instruction can be used to reverse the order of ASCII-code characters in each word.
- SWAP(637) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

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.

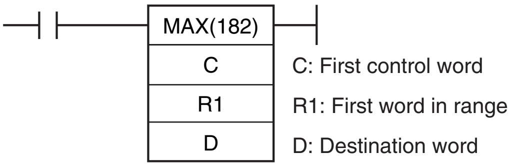
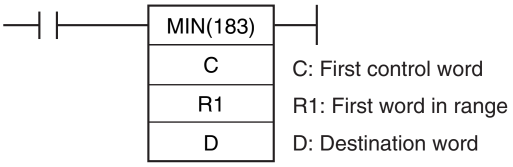


	15		8 7	0
W0	4	1	4	2
W1	4	3	4	4
W2	4	5	4	6
to	to			
W9	3	0	3	1

	15		8 7	0
W0	4	2	4	1
W1	4	4	4	3
W2	4	6	4	5
to	to			
W9	3	1	3	0

MAX/MIN

Instruction	Mnemonic	Variations	Function code	Function
FIND MAXIMUM	MAX	@MAX	182	Finds the maximum value in the range.
FIND MINIMUM	MIN	@MIN	183	Finds the minimum value in the range.

	MAX	MIN
Symbol		

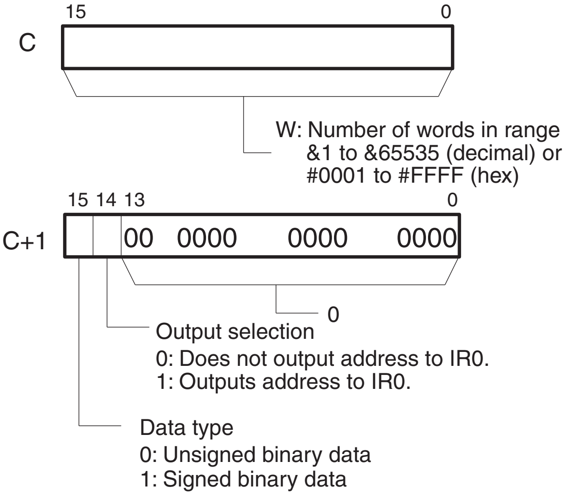
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

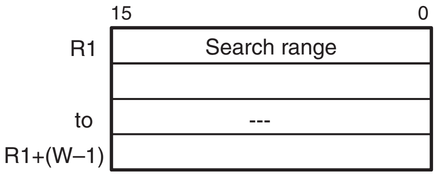
Operands

Operand	Description	Data type	Size
C	First control word	UDINT	2
R1	First word in range	UINT	Variable
D	Destination word	UINT	1

C: First control word



R1: First word in range



Note C and C+1, R1 and R1+(W-1) must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con- stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D												OK						

Flags

Name	Label	Operation	
		MAX	MIN
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 0. • OFF in all other cases.	• ON if the minimum value is 0. • 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.	• ON if bit 15 is ON in the word containing the minimum value. • OFF in all other cases.

Related Auxiliary Area Words and Bits

Name	Address	Operation
IR0 Output for Background Execution	A595 and A596	When an index register is specified as the output for an instruction processed in the background, A595 and A596 receive the output instead of IR0. (A595 contains the rightmost digits, and A596 contains the leftmost digits.)
ER/AER Flag for Background Execution	A395.10	This flag is turned ON if an error or illegal access occurs during background execution.

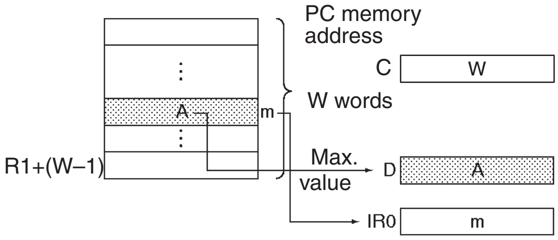
Function

● MAX

MAX(182) searches the range of memory from R1 to R1+(W-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.

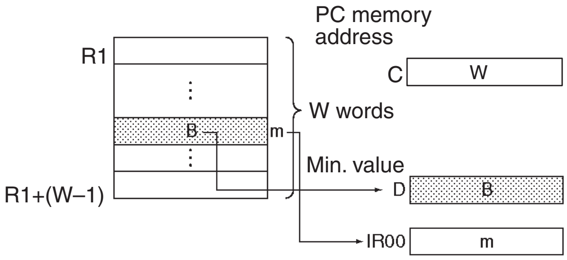


● MIN

MIN(183) searches the range of memory from R1 to R1+(W-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.



Hint

- 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.
- MAX(182)/MIN(183) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

● MAX

If background execution is enabled in the PLC Setup, the PLC memory address of the word containing the maximum value will be output to Auxiliary Area words A595 and A596 instead of IR0.

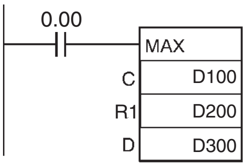
● MIN

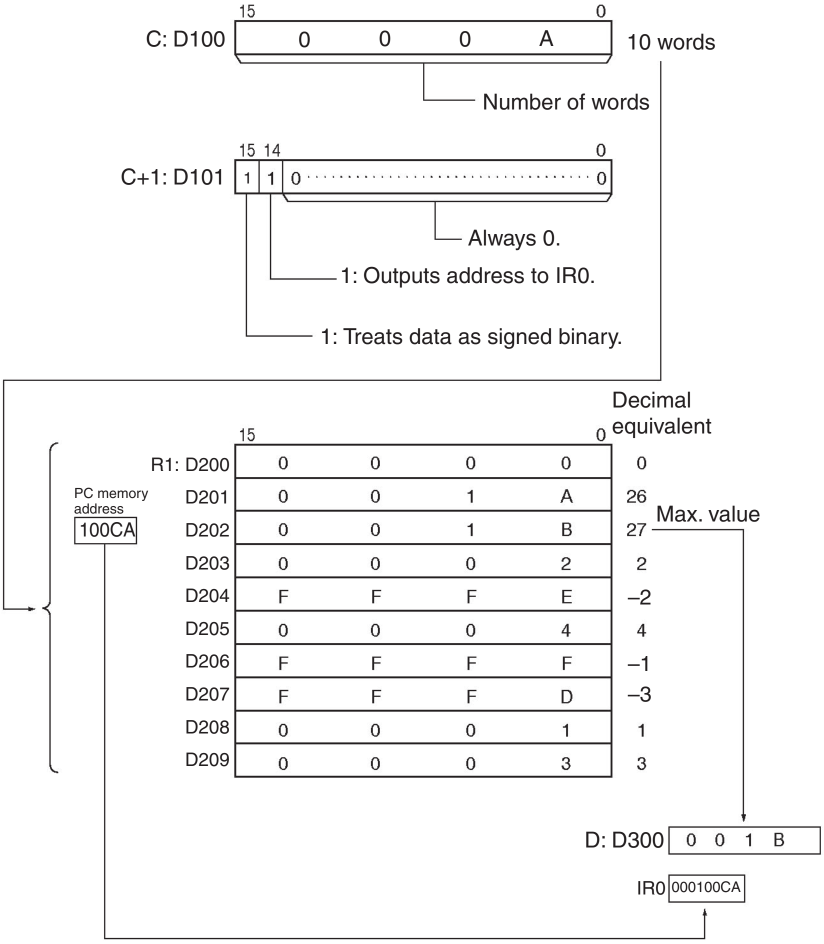
If background execution is enabled in the PLC Setup, the PLC memory address of the word containing the minimum value will be output to Auxiliary Area words A595 and A596 instead of IR0.

Example Programming

● MAX

When CIO 0.00 turns ON in the following example, MAX(182) searches the 10-word range 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.

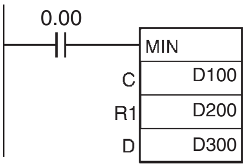


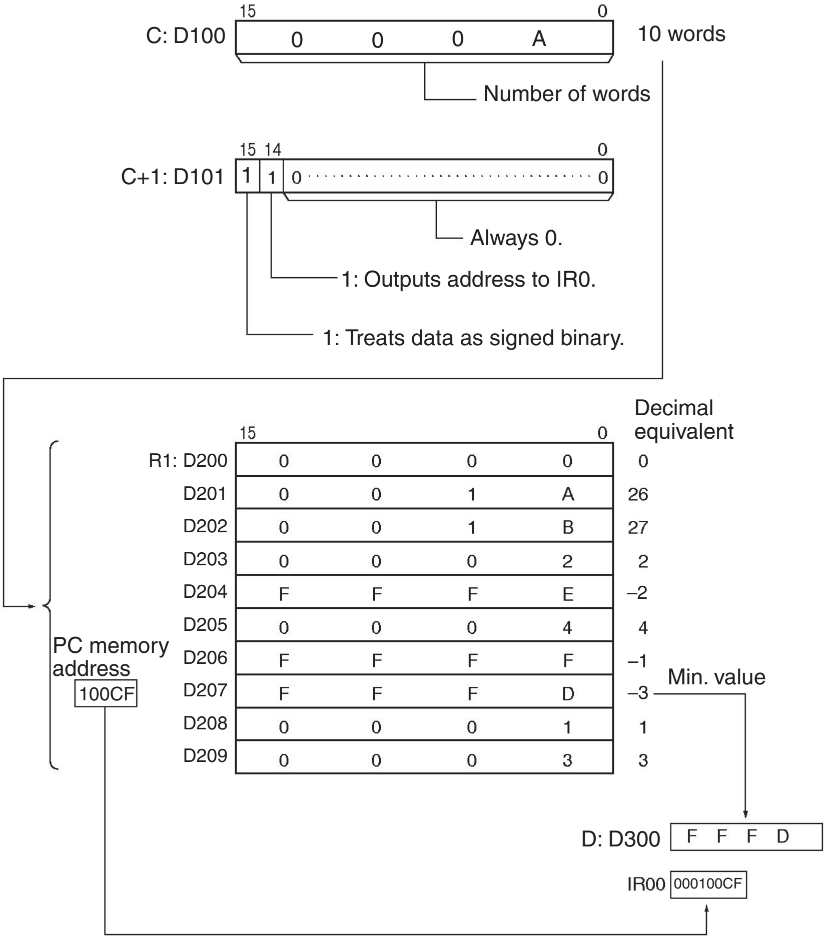


	15			0	Decimal equivalent
R1: D200	0	0	0	0	0
D201	0	0	1	A	26
D202	0	0	1	B	27
D203	0	0	0	2	2
D204	F	F	F	E	–2
D205	0	0	0	4	4
D206	F	F	F	F	–1
D207	F	F	F	D	–3
D208	0	0	0	1	1
D209	0	0	0	3	3

● MIN

When CIO 0.00 turns ON in the following example, MIN(183) searches the 10-word range 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.

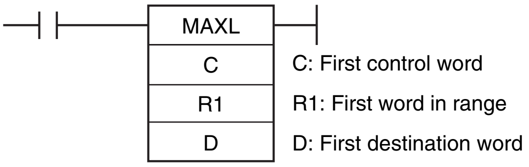




	15			0	Decimal equivalent
R1: D200	0	0	0	0	0
D201	0	0	1	A	26
D202	0	0	1	B	27
D203	0	0	0	2	2
D204	F	F	F	E	–2
D205	0	0	0	4	4
D206	F	F	F	F	–1
D207	F	F	F	D	–3
D208	0	0	0	1	1
D209	0	0	0	3	3

MAXL

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE FIND MAXIMUM	MAXL	@MAXL	174	Treats the specified number of data items as double word table data and outputs the maximum value in the table.

	MAXL
Symbol	

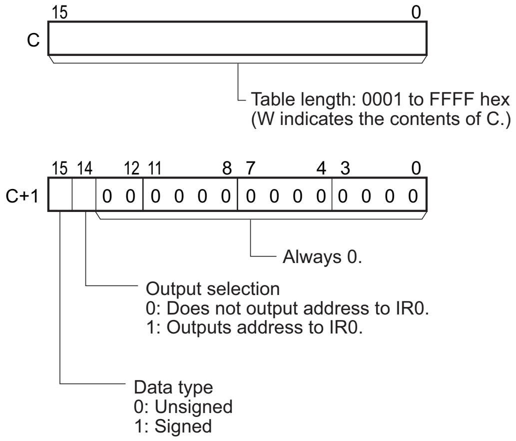
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	UDINT	2
R1	First word in range	UDINT	Variable
D	First destination word	UDINT	2

C: First control word



R1: First word in range

	15 0
R1	Lower 16 bits of value 1
R1+1	Upper 16 bits of value 1
R1+2	Lower 16 bits of value 2
R1+3	Upper 16 bits of value 2
to	to
R1+(2W-2)	Lower 16 bits of value n
R1+(2W-1)	Upper 16 bits of value n

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

Flags

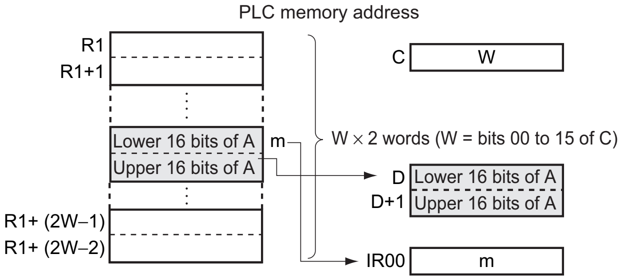
Name	Label	Operation
Error Flag	P_ER	• ON if the length of the table specified in C (bits 00 to 15) is not between 0001 and FFFF hex. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the maximum value is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if bit 15 of D+1 is ON. • OFF in all other cases.

Function

MAXL(174) searches the range of memory from R1 to R1+(2W-1) for the maximum double word value in the range and outputs that maximum value to D and D+1.

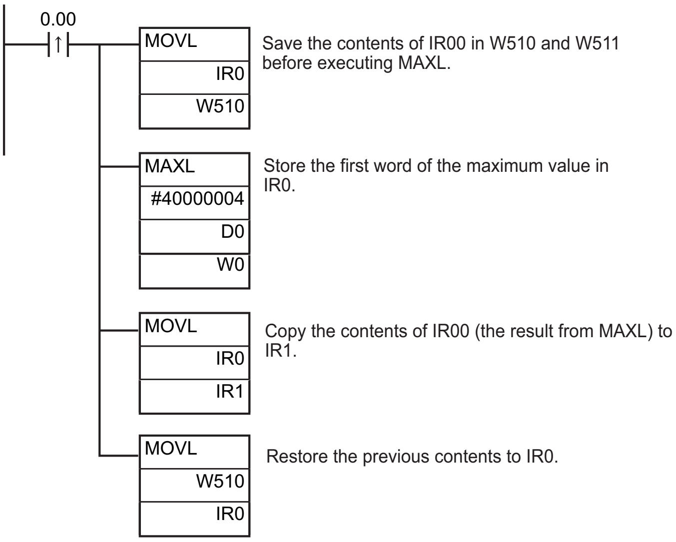
When bit 14 of C+1 has been set to 1, MAXL(174) writes the PLC memory address of the lower word of the words containing the maximum value to IR0. (If two or more double words within the range contain the same maximum value, the address of lower word of the first double word containing the maximum value is written to IR0.)

When bit 15 of C+1 has been set to 1, MAXL treats the data within the range as signed binary data. (A negative value is given as it's two's complement.)



● Saving and Restoring Values in Index Registers

MAXL can be used to write the PLC memory address of the lower word of the words containing the maximum value to IR0. If IR0 is being used for other purposes in other places in the program, be sure to save and restore the contents of IR0 before and after MAXL. (This information applies to MAXF, MAXD, MINL, MINF, and MIND as well. For MINL, MINF, and MIND, the PLC memory address of the lower word of the minimum value would be written to IR0.)



Save the contents of IR00 in W510 and W511 before executing MAXL.

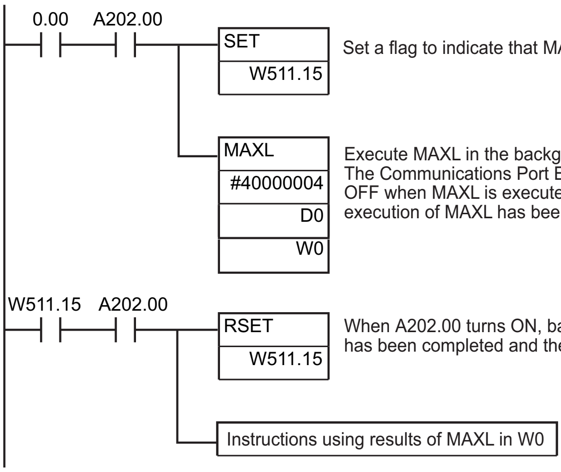
Store the first word of the maximum value in IR0.

Copy the contents of IR00 (the result from MAXL) to IR1.

Restore the previous contents to IR0.

● Background Processing

The execution time of MAXL will increase proportionately to the size of the table data. Background processing can be used for MAXL to suppress variations in the cycle time of the CPU Unit when the table data is too large. (This information applies to MAXF, MAXD, MINL, MINF, and MIND as well.)



Set a flag to indicate that MAXL is being executed.

Execute MAXL in the background.
The Communications Port Enabled Flag (A202.00) will turn OFF when MAXL is executed. A202.00 will turn ON when execution of MAXL has been completed.

When A202.00 turns ON, background processing for MAXL has been completed and the result stored in W0 can be used.

Example Programming

When CIO 0.00 turns ON in the following example, a table of four double word values starting at D0 is searched for the maximum value. The maximum value is stored in W0 and W1.

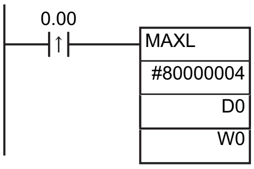
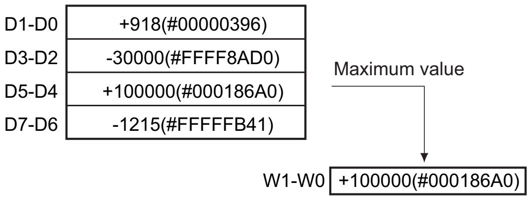


Table length: 4

Signed

PLC memory address not output to IR0



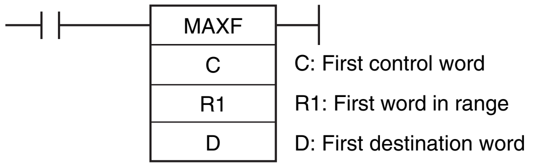
D1-D0	+918(#00000396)
D3-D2	-30000(#FFFF8AD0)
D5-D4	+100000(#000186A0)
D7-D6	-1215(#FFFFFB41)

Maximum value

W1-W0 +100000(#000186A0)

MAXF

Instruction	Mnemonic	Variations	Function code	Function
FIND MAXIMUM FLOATING	MAXF	@MAXF	176	Treats the specified number of data items as a table of single-precision floating-point data and outputs the maximum value in the table.

	MAXF
Symbol	

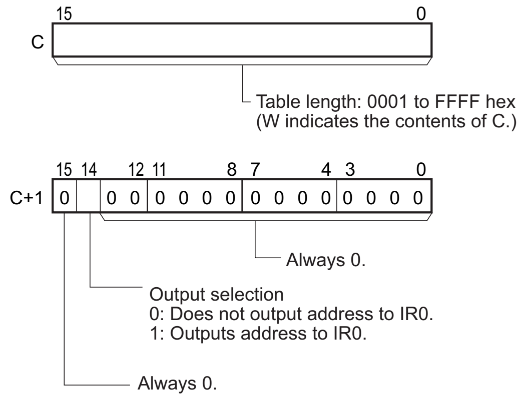
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	DWORD	2
R1	First word in range	REAL	Variable
D	First destination word	REAL	2

C: First control word



R1: First word in range

	15 0
R1	Lower 16 bits of value 1
R1+1	Upper 16 bits of value 1
R1+2	Lower 16 bits of value 2
R1+3	Upper 16 bits of value 2
to	to
R1+(2W–2)	Lower 16 bits of value n
R1+(2W–1)	Upper 16 bits of value n

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

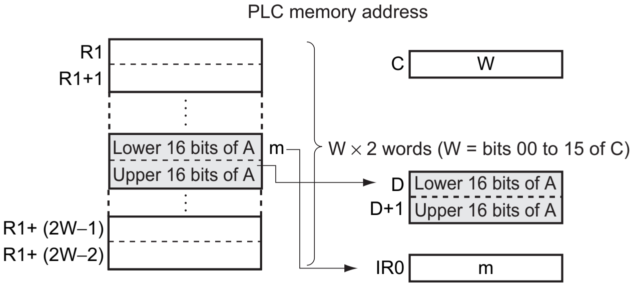
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the length of the table specified in C (bits 00 to 15) is not between 0001 and FFFF hex. • ON if the table data is nonnumeric. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the maximum value is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if the maximum value is negative. • OFF in all other cases.

Function

MAXF(176) searches the range of memory from R1 to R1+(2W-1) for the maximum single-precision floating-point value (32-bit according to IEEE 754) in the range and outputs that maximum value to D and D+1.

When bit 14 of C+1 has been set to 1, MAXF(176) writes the PLC memory address of the lower word of the words containing the maximum value to IR0. (If two or more double words within the range contain the same maximum value, the address of lower word of the first double word containing the maximum value is written to IR0.)

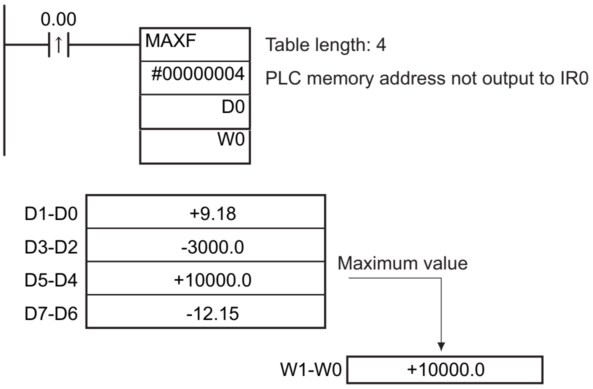


Hint

Refer to the description of MAXL(174) for information on saving and restoring values in Index Registers and information on background processing.

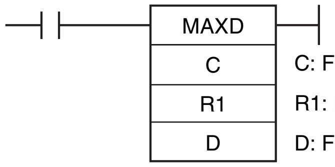
Example Programming

When CIO 0.00 turns ON in the following example, a table of four single-precision floating-point values starting at D0 is searched for the maximum value.
The maximum value is stored in W0 and W1.



MAXD

Instruction	Mnemonic	Variations	Function code	Function
FIND DOUBLE MAXIMUM FLOATING	MAXD	@MAXD	178	Treats the specified number of data items as a table of double-precision floating-point data and outputs the maximum value in the table.

	MAXD
Symbol	 C: First control word R1: First word in range D: First destination word

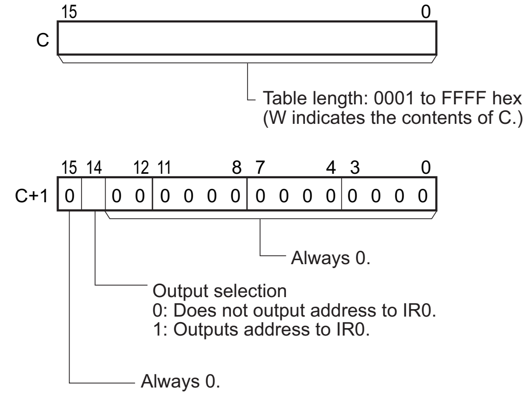
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	DWORD	2
R1	First word in range	LREAL	Variable
D	First destination word	LREAL	4

C: First control word



R1: First word in range

R1	Lowest 16 bits of value 1
R1+1	Lower 16 bits of value 1
R1+2	Upper 16 bits of value 1
R1+3	Highest 16 bits of value 1
R1+4	Lowest 16 bits of value 2
R1+5	Lower 16 bits of value 2
R1+6	Upper 16 bits of value 2
R1+7	Highest 16 bits of value 2
to	to
R1+(4W–4)	Lowest 16 bits of value n
R1+(4W–3)	Lower 16 bits of value n
R1+(4W–2)	Upper 16 bits of value n
R1+(4W–1)	Highest 16 bits of value n

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

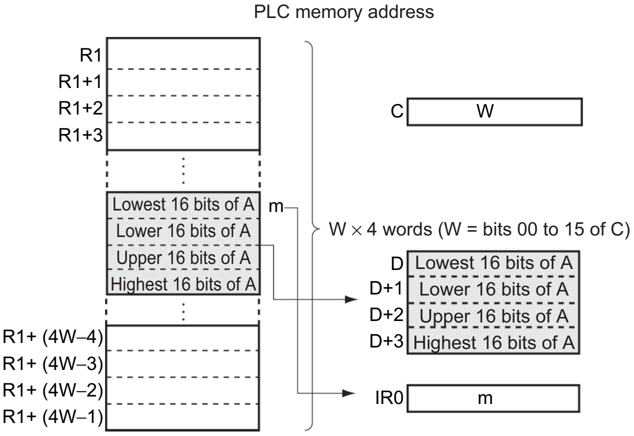
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the length of the table specified in C (bits 00 to 15) is not between 0001 and FFFF hex. • ON if the table data is nonnumeric. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the maximum value is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if the maximum value is negative. • OFF in all other cases.

Function

MAXD(178) searches the range of memory from R1 to R1+(4W-1) for the maximum double-precision floating-point value (64-bit according to IEEE 754) in the range and outputs that maximum value to D to D+3.

When bit 14 of C+1 has been set to 1, MAXD(178) writes the PLC memory address of the lower word of the words containing the maximum value to IR0. (If two or more double words within the range contain the same maximum value, the address of lower word of the first double word containing the maximum value is written to IR0.)

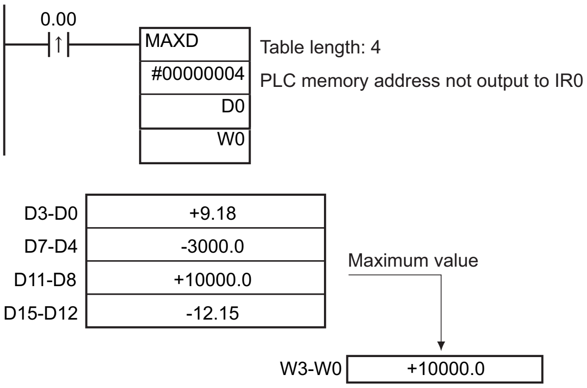


Hint

Refer to the description of MAXL(174) for information on saving and restoring values in Index Registers and information on background processing.

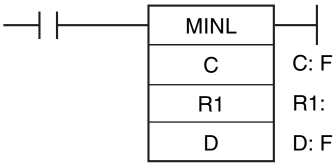
Example Programming

When CIO 0.00 turns ON in the following example, a table of four double-precision floating-point values starting at D0 is searched for the maximum value.
The maximum value is stored in W0 to W3.



MINL

Instruction	Mnemonic	Variations	Function code	Function
DOUBLE FIND MINIMUM	MINL	@MINL	175	Treats the specified number of data items as double word table data and outputs the minimum value in the table.

	MINL
Symbol	 C: First control word R1: First word in range D: First destination word

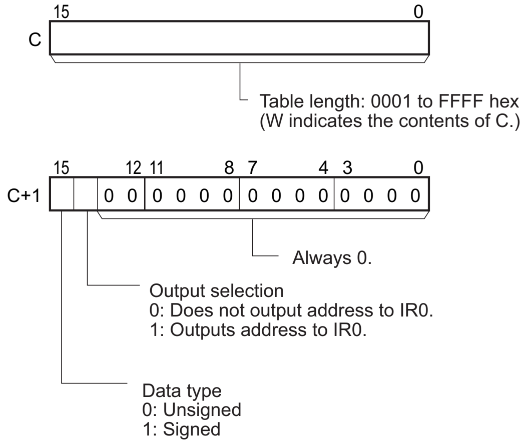
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	UDINT	2
R1	First word in range	UDINT	Variable
D	First destination word	UDINT	2

C: First control word



R1: First word in range

	15 0
R1	Lower 16 bits of value 1
R1+1	Upper 16 bits of value 1
R1+2	Lower 16 bits of value 2
R1+3	Upper 16 bits of value 2
to	to
R1+(2W–2)	Lower 16 bits of value n
R1+(2W–1)	Upper 16 bits of value n

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

Flags

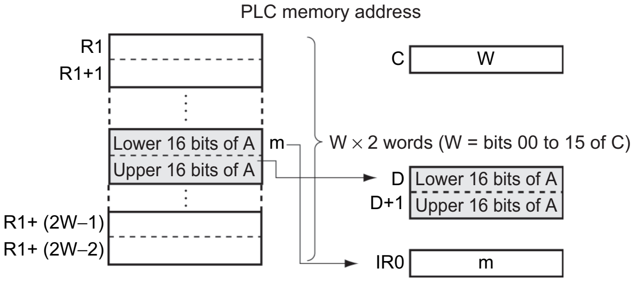
Name	Label	Operation
Error Flag	P_ER	• ON if the length of the table specified in C (bits 00 to 15) is not between 0001 and FFFF hex. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the minimum value is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if bit 15 of D+1 is ON. • OFF in all other cases.

Function

MINL(175) searches the range of memory from R1 to R1+(2W-1) for the minimum double word value in the range and outputs that minimum value to D and D+1.

When bit 14 of C+1 has been set to 1, MINL(175) writes the PLC memory address of the lower word of the words containing the minimum value to IR0. (If two or more double words within the range contain the same minimum value, the address of lower word of the first double word containing the minimum value is written to IR0.)

When bit 15 of C+1 has been set to 1, MINL(175) treats the data within the range as signed binary data. (A negative value is given as it's two's complement.)

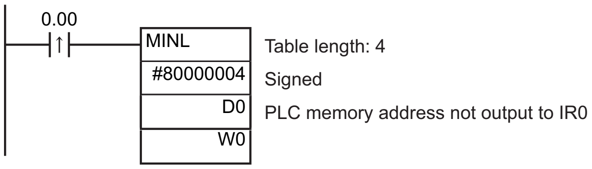


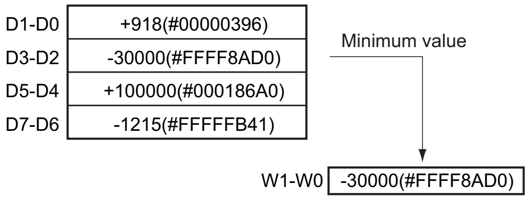
Hint

Refer to the description of MAXL(174) for information on saving and restoring values in Index Registers and information on background processing.

Example Programming

When CIO 0.00 turns ON in the following example, a table of four double word values starting at D0 is searched for the minimum value.
The minimum value is stored in W0 and W1.



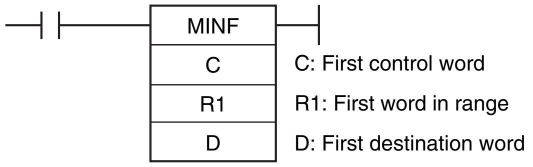


MINF

Instruction	Mnemonic	Variations	Function code	Function
FIND MINIMUM FLOATING	MINF	@MINF	177	Treats the specified number of data items as a table of single-precision floating-point data and outputs the minimum value in the table.

Symbol

MINF



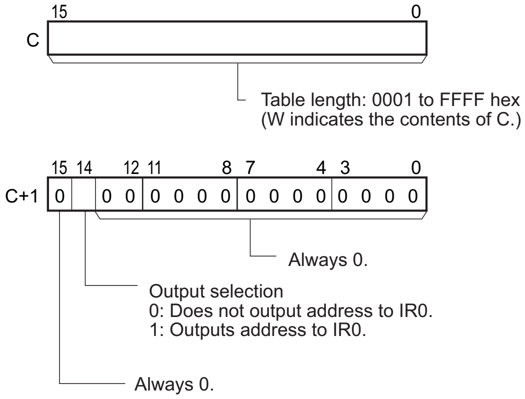
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	DWORD	2
R1	First word in range	REAL	Variable
D	First destination word	REAL	2

C: First control word



R1: First word in range

	15 0
R1	Lower 16 bits of value 1
R1+1	Upper 16 bits of value 1
R1+2	Lower 16 bits of value 2
R1+3	Upper 16 bits of value 2
to	to
R1+(2W–2)	Lower 16 bits of value n
R1+(2W–1)	Upper 16 bits of value n

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

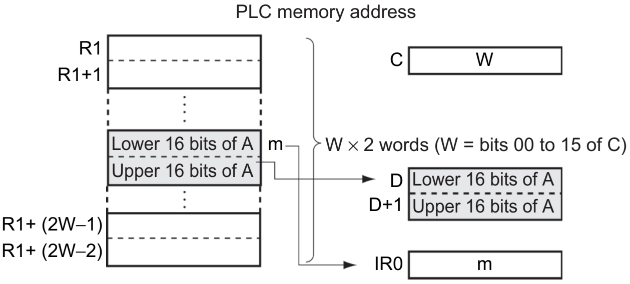
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the length of the table specified in C (bits 00 to 15) is not between 0001 and FFFF hex. • ON if the table data is nonnumeric. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the minimum value is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if the minimum value is negative. • OFF in all other cases.

Function

MINF(177) searches the range of memory from R1 to R1+(2W-1) for the minimum single-precision floating-point value (32-bit according to IEEE 754) in the range and outputs that minimum value to D and D+1.

When bit 14 of C+1 has been set to 1, MINF(177) writes the PLC memory address of the lower word of the words containing the minimum value to IR0. (If two or more double words within the range contain the same minimum value, the address of lower word of the first double word containing the minimum value is written to IR0.)

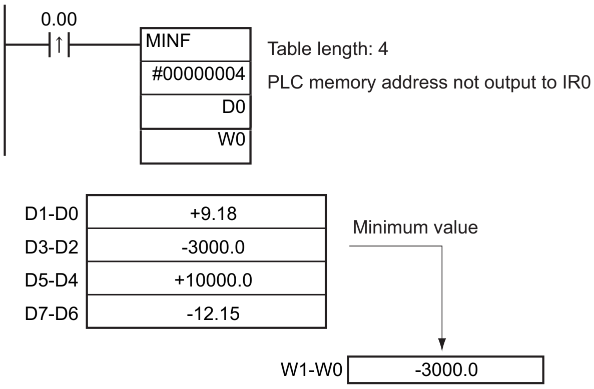


Hint

Refer to the description of MAXL(174) for information on saving and restoring values in Index Registers and information on background processing.

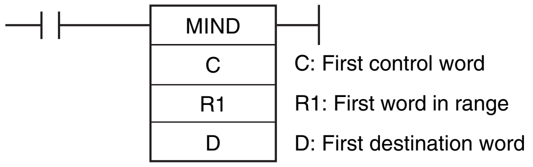
Example Programming

When CIO 0.00 turns ON in the following example, a table of four single-precision floating-point values starting at D0 is searched for the minimum value.
The minimum value is stored in W0 and W1.



MIND

Instruction	Mnemonic	Variations	Function code	Function
FIND DOUBLE MINIMUM FLOATING	MIND	@MIND	179	Treats the specified number of data items as a table of double-precision floating-point data and outputs the minimum value in the table.

	MIND
Symbol	

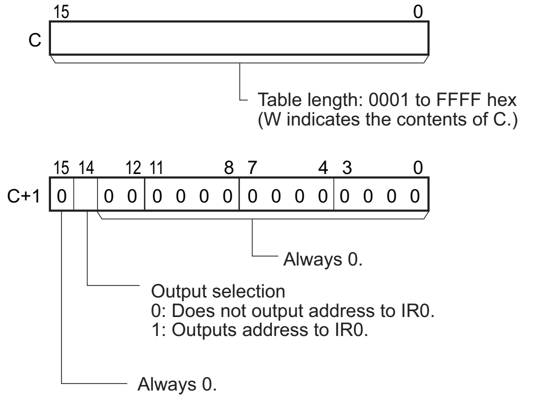
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

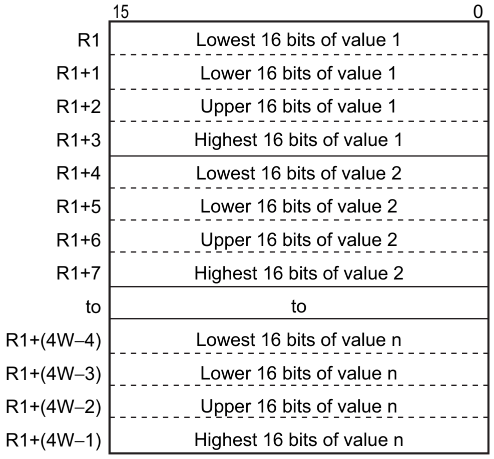
Operands

Operand	Description	Data type	Size
C	First control word	DWORD	2
R1	First word in range	LREAL	Variable
D	First destination word	LREAL	4

C: First control word



R1: First word in range



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

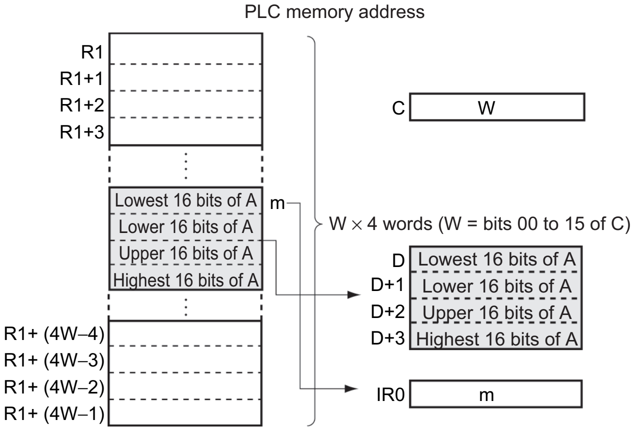
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the length of the table specified in C (bits 00 to 15) is not between 0001 and FFFF hex. • ON if the table data is nonnumeric. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the minimum value is 0. • OFF in all other cases.
Negative Flag	P_N	• ON if the minimum value is negative. • OFF in all other cases.

Function

MIND(179) searches the range of memory from R1 to R1+(4W-1) for the minimum double-precision floating-point value (64-bit according to IEEE 754) in the range and outputs that minimum value to D to D+3.

When bit 14 of C+1 has been set to 1, MIND(179) writes the PLC memory address of the lower word of the words containing the minimum value to IR0. (If two or more double words within the range contain the same minimum value, the address of lower word of the first double word containing the minimum value is written to IR0.)

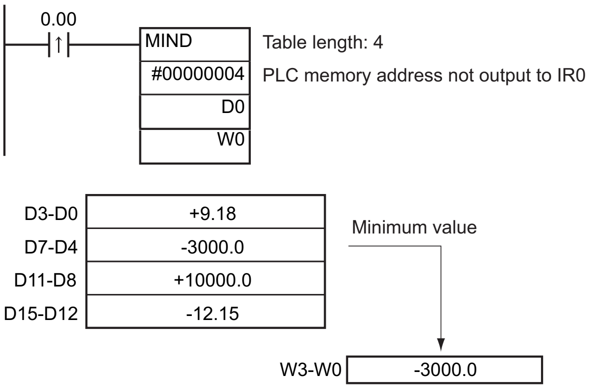


Hint

Refer to the description of MAXL(174) for information on saving and restoring values in Index Registers and information on background processing.

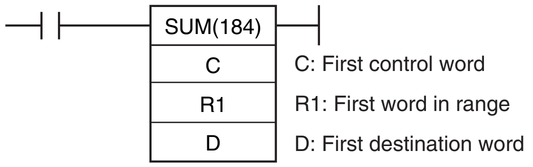
Example Programming

When CIO 0.00 turns ON in the following example, a table of four double-precision floating-point values starting at D0 is searched for the minimum value.
The minimum value is stored in W0 to W3.



SUM

Instruction	Mnemonic	Variations	Function code	Function
SUM	SUM	@SUM	184	Adds the bytes or words in the range and outputs the result to two words.

	SUM
Symbol	

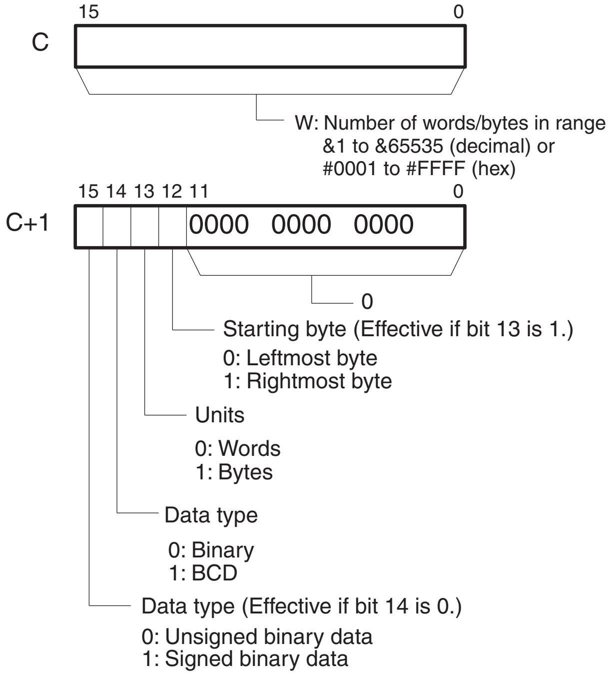
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

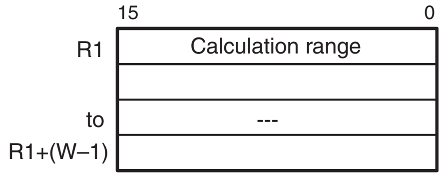
Operands

Operand	Description	Data type	Size
C	First control word	UDINT	2
R1	First word in range	UINT	Variable
D	First destination word	UDINT	2

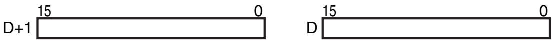
C: First control word



R1: First word in range



D: First destination word



The leftmost four digits are stored in D+1 and the rightmost four digits are stored in D.

Note C and C+1, all of the words in the calculation range must be in the same data area.

● Operand Specifications

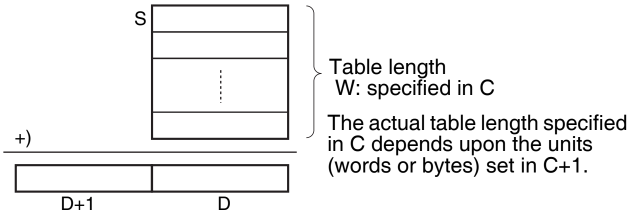
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

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 0. • OFF in all other cases.
Negative Flag	N	• ON if bit 15 is ON in the result. • OFF in all other cases.

Function

SUM(184) adds W 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 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).

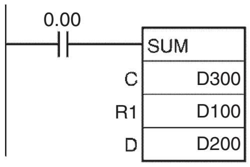
When bit 14 of C+1 has been set to 0, SUM(184) treats the data as binary. In this case, bit 15 determines whether the data is signed (bit 15 = 1) or unsigned (bit 15 = 0).

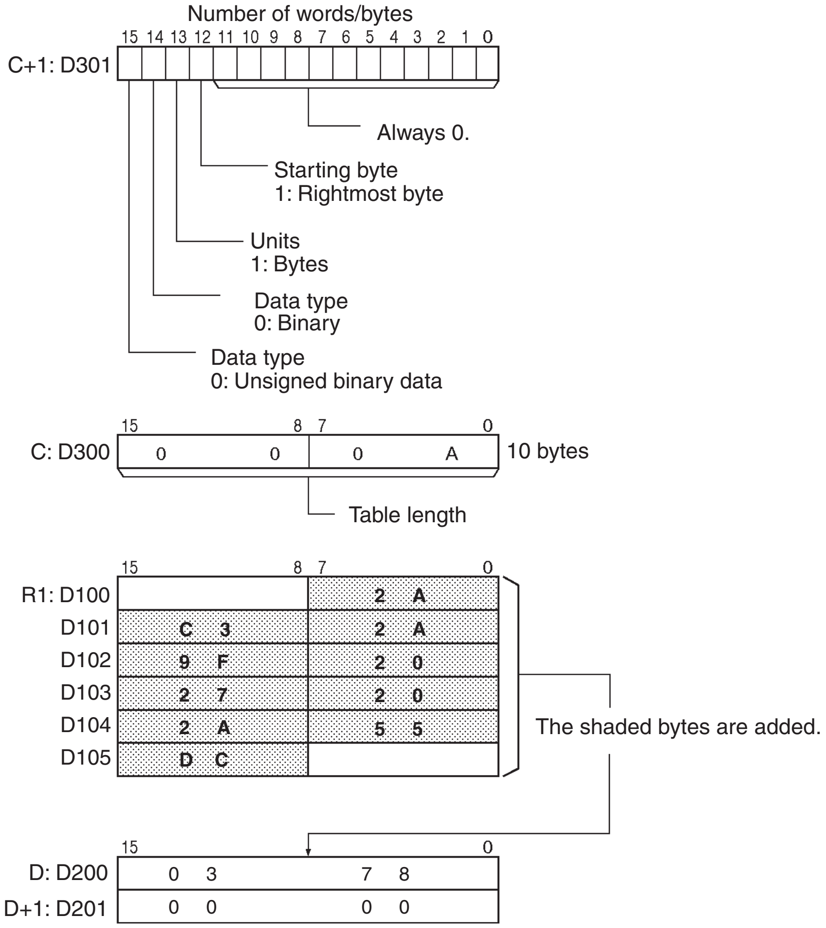
Hint

SUM(184) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

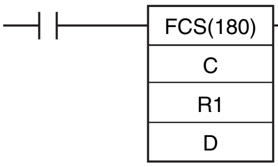
When CIO 0.00 is ON in the following example, SUM(184) adds 10 bytes of unsigned binary data beginning with the rightmost byte of D100 and writes the result to D201 and D200.





FCS

Instruction	Mnemonic	Variations	Function code	Function
FRAME CHECKSUM	FCS	@FCS	180	Calculates the FCS value for the specified range and outputs the result in ASCII.

	FCS
Symbol	 C: First control word R1: First word in range D: First destination word

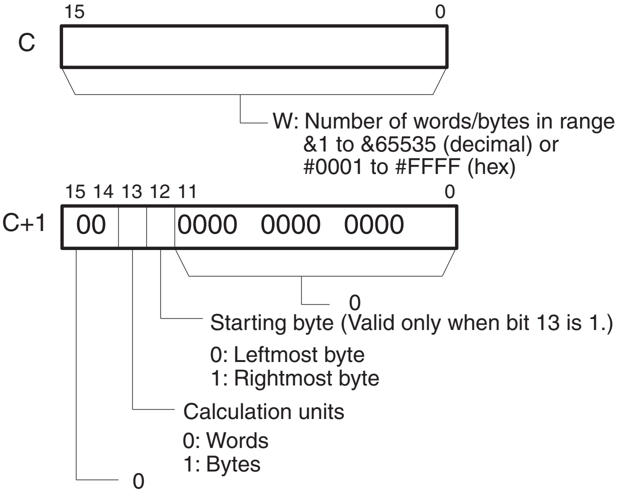
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

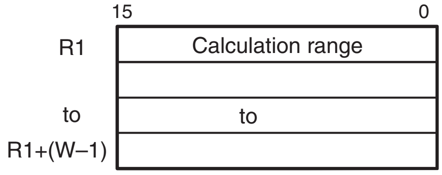
Operands

Operand	Description	Data type	Size
C	First control word	UDINT	2
R1	First word in range	UINT	Variable
D	First destination word	UINT	Variable

C: First control word

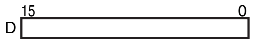


R1: First word in range

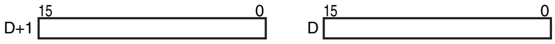


D: First destination word

When bytes are specified as the operation units:



When words are specified as the operation units:



The leftmost four digits are stored in D+1 and the rightmost four digits are stored in D.

Note C and C+1, all of the words in the calculation range must be in the same data area.

● Operand Specifications

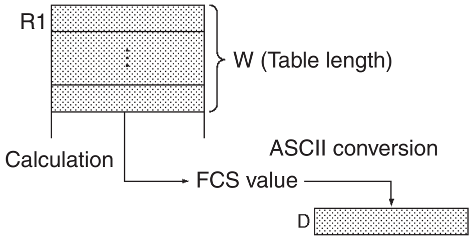
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
R1											---							
D																		

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.

Function

FCS(180) calculates the FCS value for W 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.



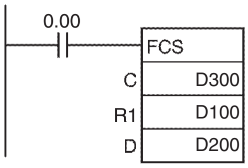
When bit 13 of C+1 has been set to 1, FCS(180) operates on bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).

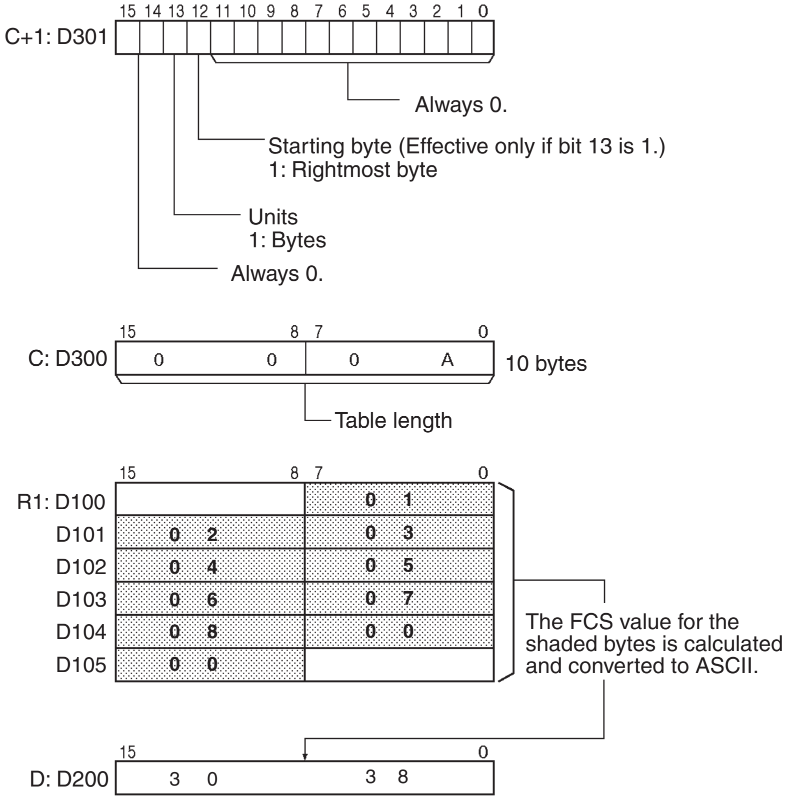
Hint

FCS(180) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

When CIO 0.00 is ON in the following example, FCS(180) calculates the FCS value for the 10 bytes of data beginning with the rightmost byte of D100 and writes the result to D200.





Tracking Instructions

Tracking Instructions

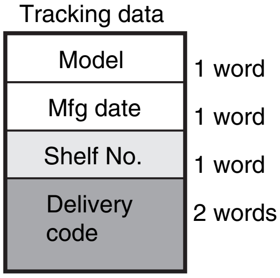
Manufacturing information on workpieces placed into a production line is generally called tracking data. Tracking data includes many pieces of information, such as the product type, model, quantity, manufacturing date, product grade, and manufacturing factory. Each group of information is called a record.

Tracking Instructions can be used to easily search for the necessary records in tracking data or to store tracking data according to the information contained in them.

The following are programming samples that use tracking instructions.

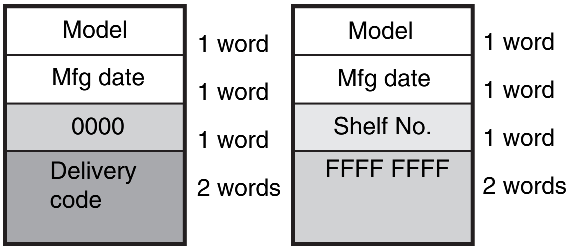
1. Finding tracking data that matches certain conditions
2. Sorting and then searching tracking data
3. Sorting tracking data that contains many records

● Sorting and Searching Shelf Numbers and Delivery Codes



A tracking data record is invalid if it contains the following.

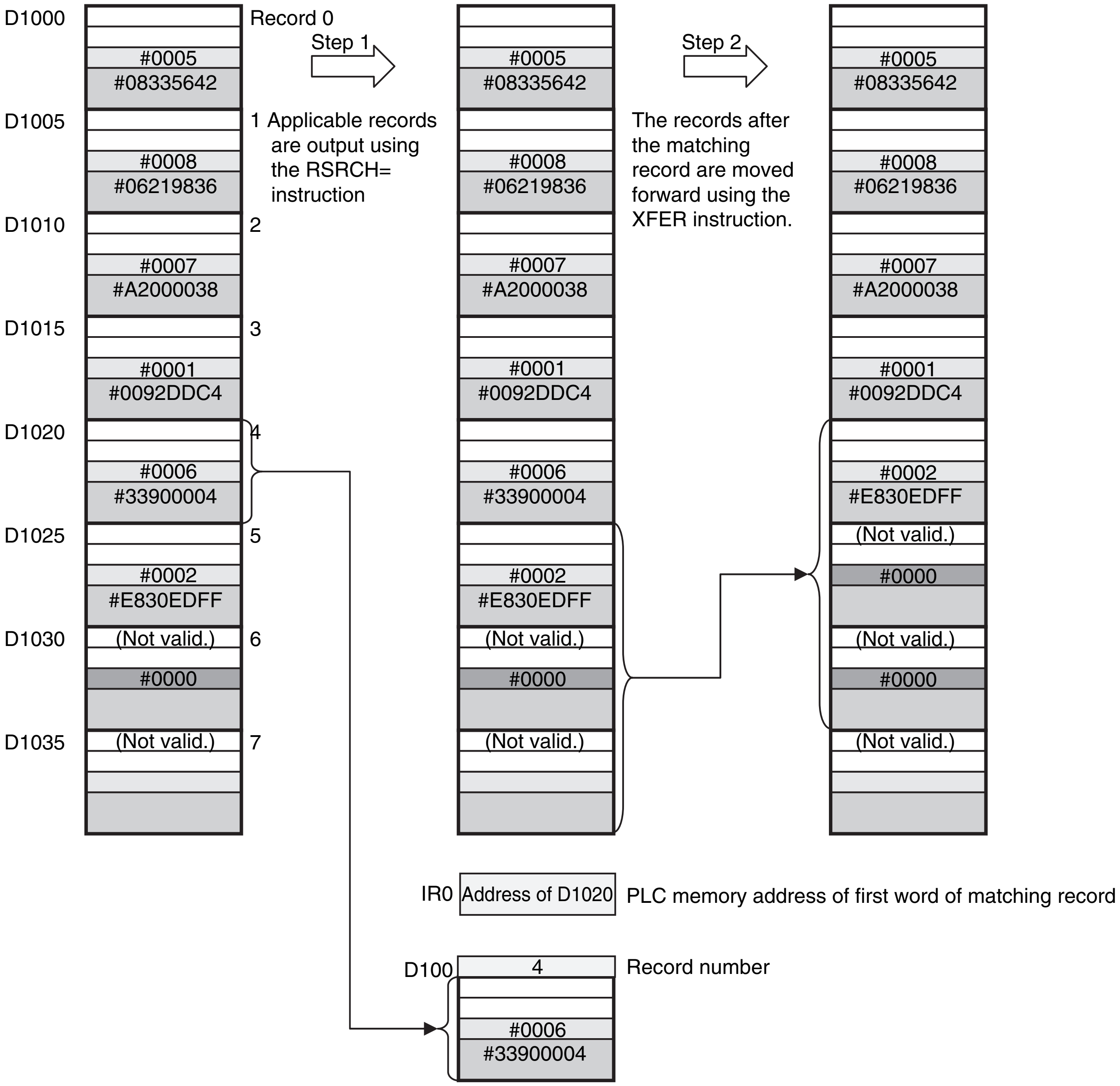
- A shelf number of 0000 hex
- A delivery code of FFFF FFFF hex



Finding Tracking Data That Matches Certain Conditions

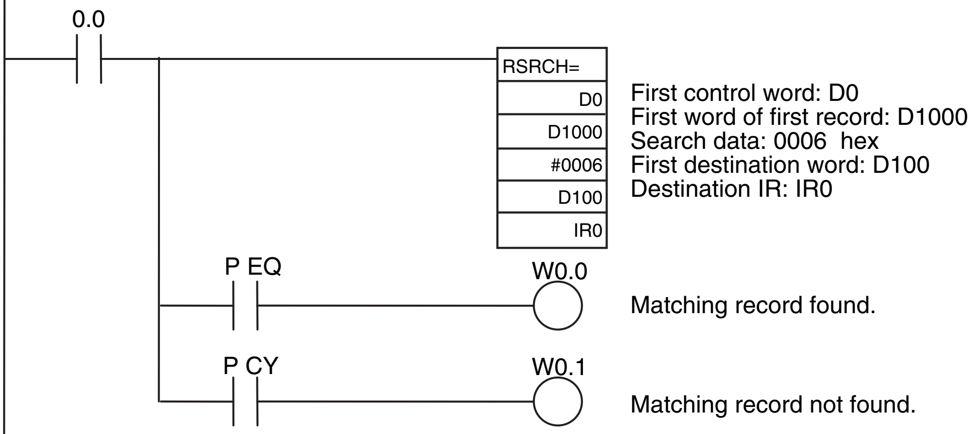
The table of tracking data is searched for shelf number 0006 hex and the data is stored in another location.

First word of first record	D1000
Record length	5
Number of records	8
First word of search results	D100
Search key	Self number
Setting to end search	Enabled for end data (shelf number = 0000 hex)
Search method	Linear search (i.e., search in order from first record)
Destination index register	IR0



C: D0	0 0 0 8	Number of records = 8
C+1: D1	0 0 0 5	Record length = 5 words
C+2: D2	0 0 0 2	Search data offset = 2 words
C+3: D3	8 0 0 1	Index register output designation: Output Search method = Linear, Setting to end search = Effective end data
C+4: D4	0 0 0 0	End data = 0000 hex
C+5: D5	0 0 0 0	Always 0000 hex.

Step 1: Corresponding records output using the RSRCH= instruction.

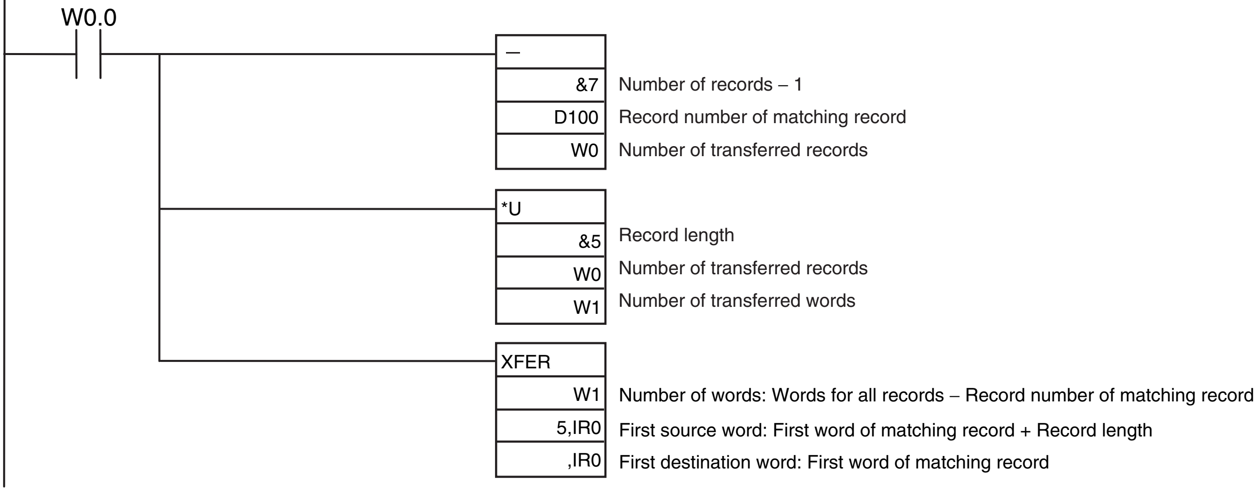


First control word: D0
First word of first record: D1000
Search data: 0006 hex
First destination word: D100
Destination IR: IR0

Matching record found.

Matching record not found.

Step 2: Records moved forward with the XFER instruction to fill in the space of the matching record.



Number of records – 1
Record number of matching record
Number of transferred records

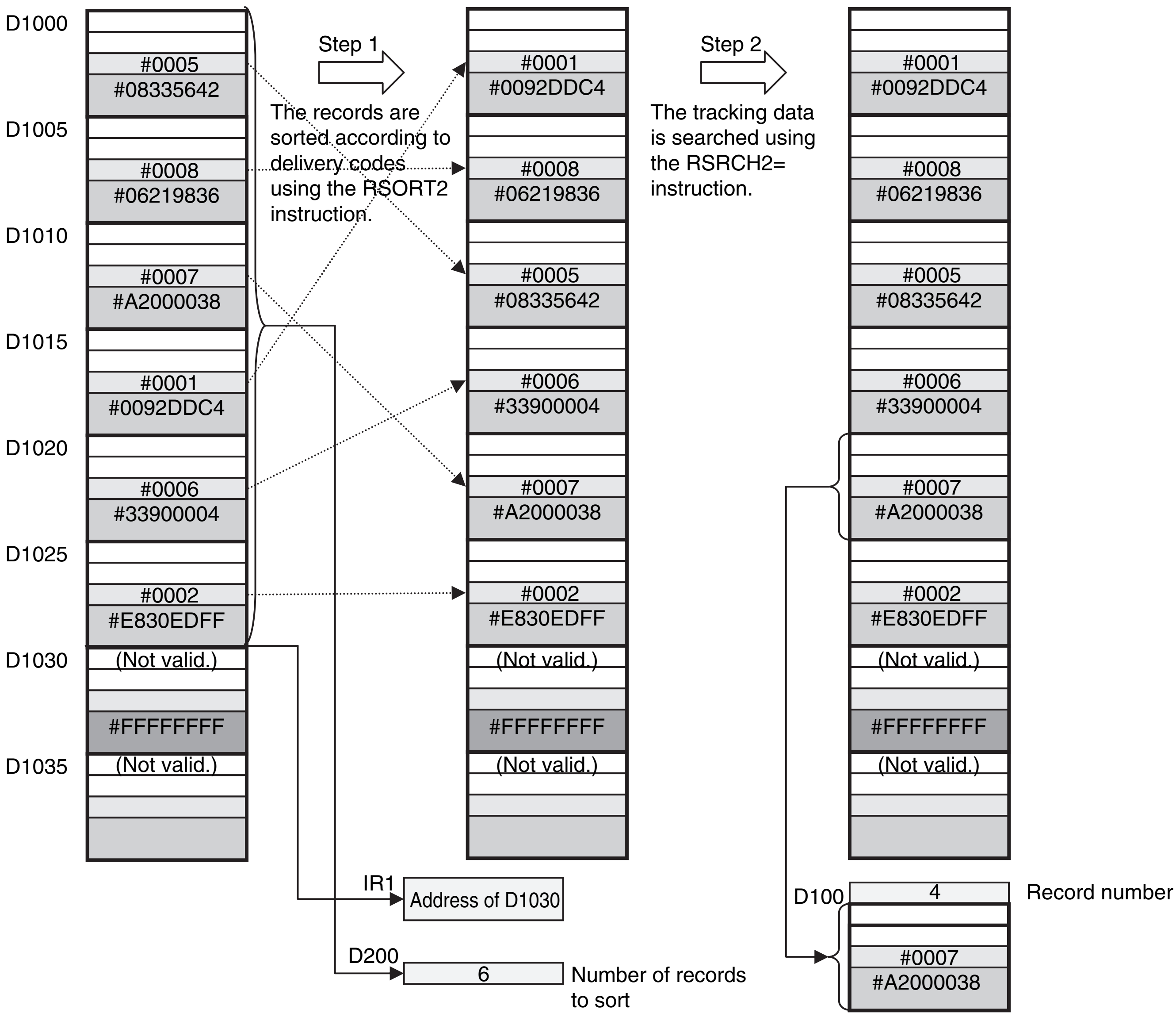
Record length
Number of transferred records
Number of transferred words

Number of words: Words for all records – Record number of matching record
First source word: First word of matching record + Record length
First destination word: First word of matching record

● Sorting and Then Searching Tracking Data

The tracking data table is sorted according to the delivery code and then the data is searched for delivery code A200 0038 hex.

First word of first record	D1000
Record length	5
Number of records	8
First word of sorting results	D200
Sort key	Delivery code
Sort options	Ascending order, Split sort disabled (complete in one cycle)
Setting to end sorting	Enabled for end data (delivery code = FFFF FFFF hex)
Sort results output to index register	IR1
First word of search results	D100
Search key	Delivery code
Setting to end search	Enabled for end PLC memory address (search results output)
Search method	Split search
Search results output to index register	Output disabled



RSORT2

Word	Contents	Setting
C: D300	0 0 0 8	Number of records = 8
C+1: D301	0 0 0 5	Record length = 5 words
C+2: D302	0 0 0 3	Sorting data offset = 3 words
C+3: D303	0 0 0 0	Split sort setting = Disabled, Ascending
C+4: D304	8 0 0 1	Index register output designation = Output, Setting to end sort = Enabled
C+5: D305	F F F F	End data = FFFF hex
C+6: D306	F F F F	End data = FFFF hex
C+7: D307 to C+86: D386	System work area	

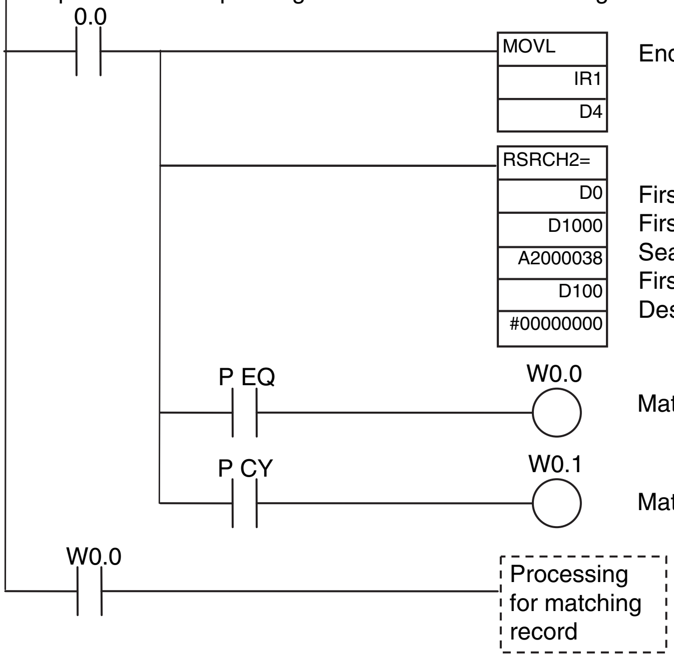
RSRCH2

Word	Contents	Setting
C: D0	0 0 0 8	Number of records = 8
C+1: D1	0 0 0 5	Record length = 5 words
C+2: D2	0 0 0 3	Search data offset = 3 words
C+3: D3	0 0 1 2	Index register output designation: Output Search method = Split search/ascending, Setting to end search = Enabled for PLC memory address
C+4: D4	0 4 0 6	PLC memory address of search end, lower word
C+5: D5	0 0 0 1	PLC memory address of search end, upper word

Step 1: The records are sorted in ascending order according to the delivery code using the RSORT2 instruction.

First control word: D300
First word of first record: D1000
First source results output word: D200
Designation IR: IR1

Step 2: The corresponding record is searched for using the RSRCH2= instruction with a split search.



End PLC memory address: IR1 (output by RSORT2).

First control word: D0
First word of first record: D1000
Search data: A200 0038 hex
First destination word: D100
Designation IR: Not output (0000 0000 hex)

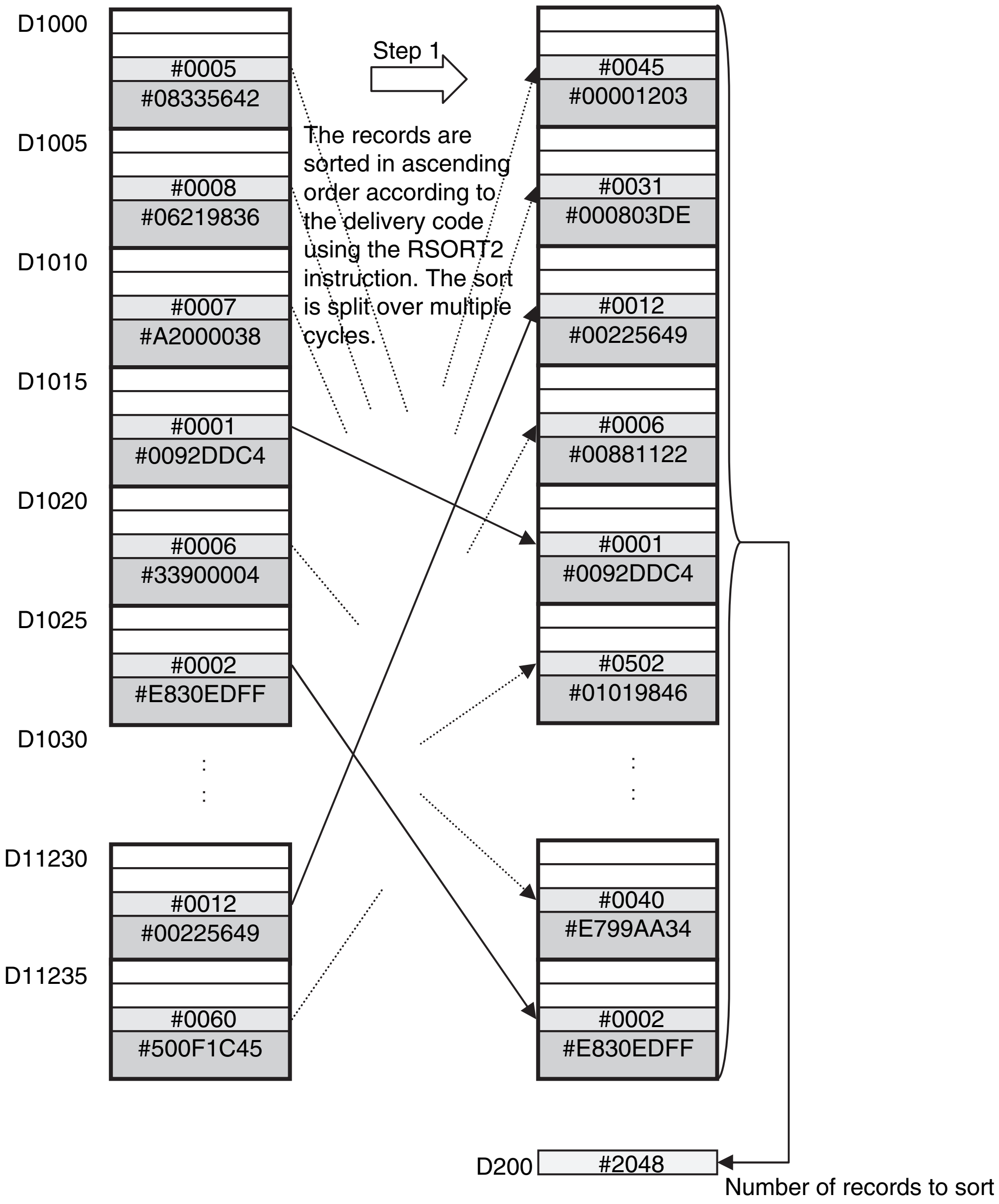
Matching record found.

Matching record not found.

Sorting Tracking Data That Contains Many Records Using a Split Sort

Tracking data that contains 2,048 records is sorted according to the delivery code. To shorten the cycle time, the sort is executed over multiple cycles.

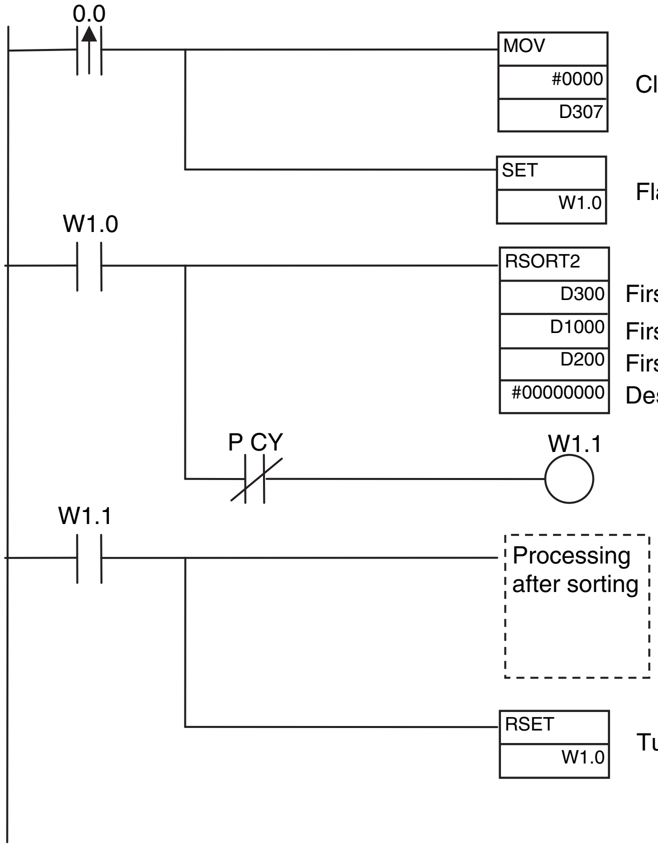
First word of first record	D1000
Record length	5
Number of records	2048
First word of sorting results	D200
Sort key	Delivery code
Sort options	Ascending, Split sort
Setting to end sorting	Disabled
Sort results output to index register	Disabled



The records are sorted in ascending order according to the delivery code using the RSORT2 instruction. The sort is split over multiple cycles.

C: D300	2 0 4 8	Number of records = 2,048
C+1: D301	0 0 0 5	Record length = 5 words
C+2: D302	0 0 0 3	Sorting data offset = 3 words
C+3: D303	8 0 0 0	Split sort setting = Enabled, Ascending
C+4: D304	0 0 0 0	Index register output designation = Disabled, Setting to end sort = Disabled
C+5: D305	0 0 0 0	Always 0000 hex.
C+6: D306	0 0 0 0	Always 0000 hex.

Step 1: The records are sorted in ascending order according to the delivery code using the RSORT2 instruction.



Clear first word of system work area (C+7).

Flag to hold ON which executing the sort.

First control word: D300

First word of first record: D1000

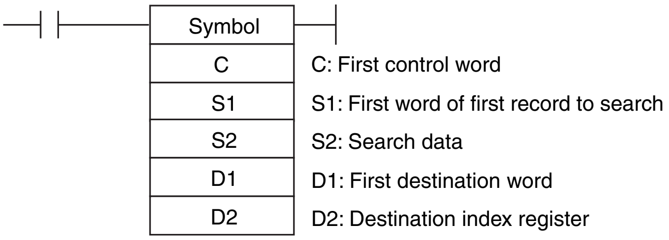
First source results output word: D200

Designation IR: None (0000 0000 hex)

Turns OFF flag that was held ON during the sort

RSRCH<, RSRCH<=, RSRCH=, RSRCH>, RSRCH>=

Instruction	Mnemonic	Variations	Function code	Function
Unsigned One-word Record Search Instructions	RSRCH< RSRCH<= RSRCH= RSRCH> RSRCH>=	@RSRCH< @RSRCH<= @RSRCH= @RSRCH> @RSRCH>=	360 361 362 363 364	An Unsigned One-word Record Search Instruction searches the specified table of records for a record with the specific data (1 word) in the specified location. When a record matching the specified condition is found, its record number and data are output.

	RSRCH<, RSRCH<=, RSRCH=, RSRCH>, RSRCH>=
Symbol	

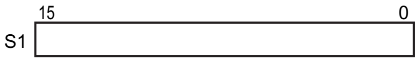
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

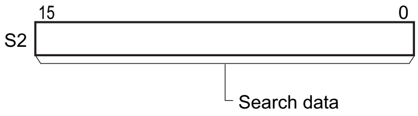
Operands

Operand	Description	Data type	Size
C	First control word	WORD	6
S1	First word of first record to search	WORD	Variable
S2	Search data	WORD	1
D1	First destination word	WORD	Variable
D2	Destination index register	DWORD	2

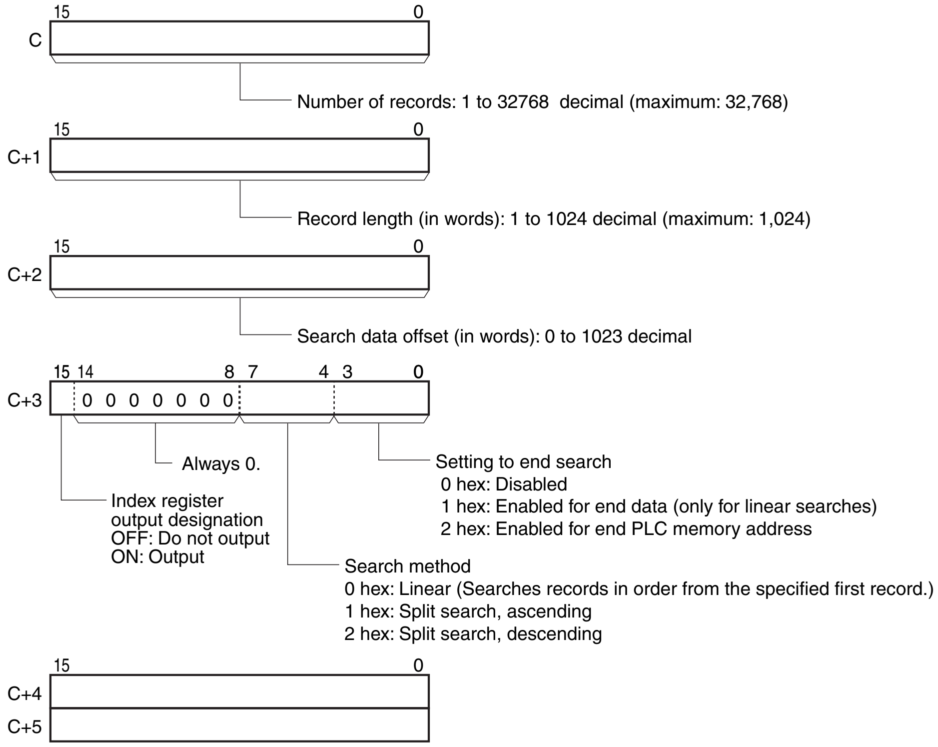
S1: First word of first record to search



S2: Search data



C: First control word



Number of records: 1 to 32768 decimal (maximum: 32,768)

Record length (in words): 1 to 1024 decimal (maximum: 1,024)

Search data offset (in words): 0 to 1023 decimal

Always 0.

Index register
output designation
OFF: Do not output
ON: Output

Setting to end search
0 hex: Disabled
1 hex: Enabled for end data (only for linear searches)
2 hex: Enabled for end PLC memory address

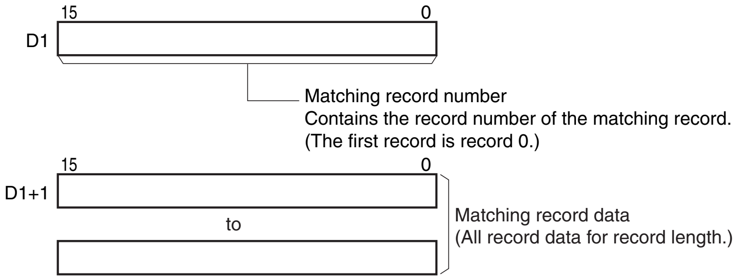
Search method
0 hex: Linear (Searches records in order from the specified first record.)
1 hex: Split search, ascending
2 hex: Split search, descending

Setting to End Search Disabled
C+4: Always 0.
C+5: Always 0.

End Data Enabled
C+4: End data (0000 to FFFF hex)
C+5: Always 0.

End PLC Memory Address Enabled
C+4: Lower word of PLC memory address
C+5: Upper word of PLC memory address

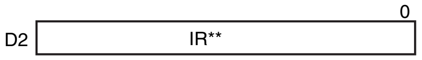
D1: First destination word



Matching record number
Contains the record number of the matching record.
(The first record is record 0.)

Matching record data
(All record data for record length.)

D2: Destination index register



The PLC memory address of the first word of the matching record is stored in the specified index register. If index register output is not specified in C+3, then 0000 0000 hex is stored in D2.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S1																		
S2											OK	OK						
D1											---	---						
D2	---	---	---	---	---	---	---	---	---	---	OK		OK	---				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON when the record length in C+1 is not between 1 and 1,024, inclusive. • ON when the number of records specified in C times the record length in C+1 is greater than 32,768. • ON when the number of records specified in C times the record length in C+1 is 0. • ON when the search data offset in C+2 is greater than one less than the record length in C+1. • OFF in all other cases.
Equals Flag	P_EQ	• ON when the search data is found. • OFF when the search data is not found.
Carry Flag	P_CY	Linear Search • ON when a matching record is found before the final record and before the condition for the setting to end the search is met. • OFF when the search is completed through the final record or until the condition for the setting to end the search is met. Split Search • Always OFF.

Function

The number of records specified by C starting from the record specified by S1 are searched for the data specified by S2. If a record that matches the search conditions is found, the Equals Flag will turn ON, the record number of the matching record will be output to D1 and the matching record data will be output starting from D1+1. If outputting to an index register is specified in C+3, the PLC memory address of the first word of the matching record is output to the specified index register. If more than one matching record is found, the record closest to the first record that is searched will be output.

If a matching record is not found, nothing will be output starting from D1 (the previous values will be maintained).

The search is performed by the method specified in C+3. There are two search methods.

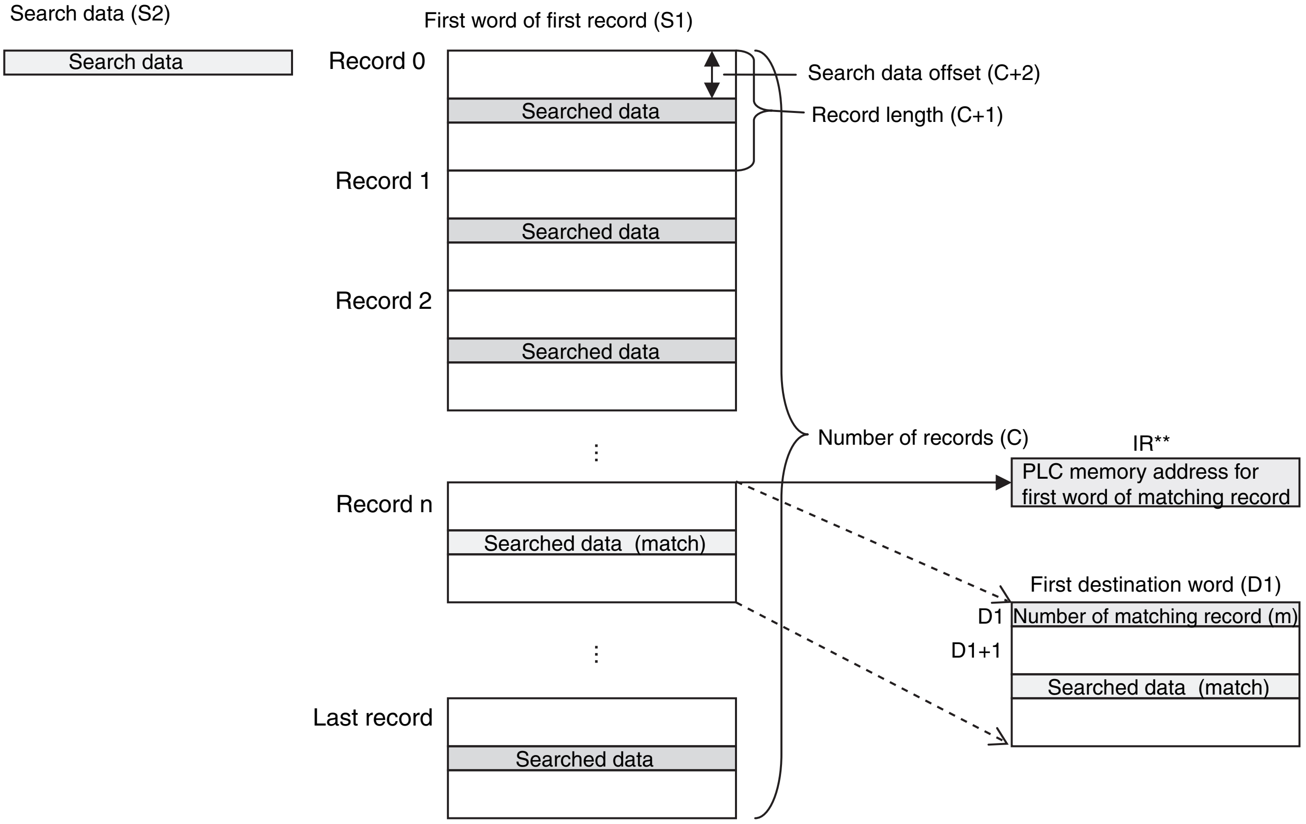
- Linear Search
  For a linear search, the records are searched in order from the specified first record. Linear searching is used when the search data has not been sorted. If a matching record is found before the maximum number of records specified in C is searched, the Carry Flag will be turned ON. (This is to indicate that there are records that were not searched.)
- Split Search (Ascending or Descending)
  If the records that are being searched have been sorted in ascending or descending order, a split search can be used to reduce the instruction execution time.

Precautions

- If the records being searched have not been sorted in ascending or descending order when a split search is specified, the results of the search will not be accurate.
- A split search cannot be specified if using search end data has been enabled. If using search end data is enabled, a linear search will be performed even if a split search is specified.

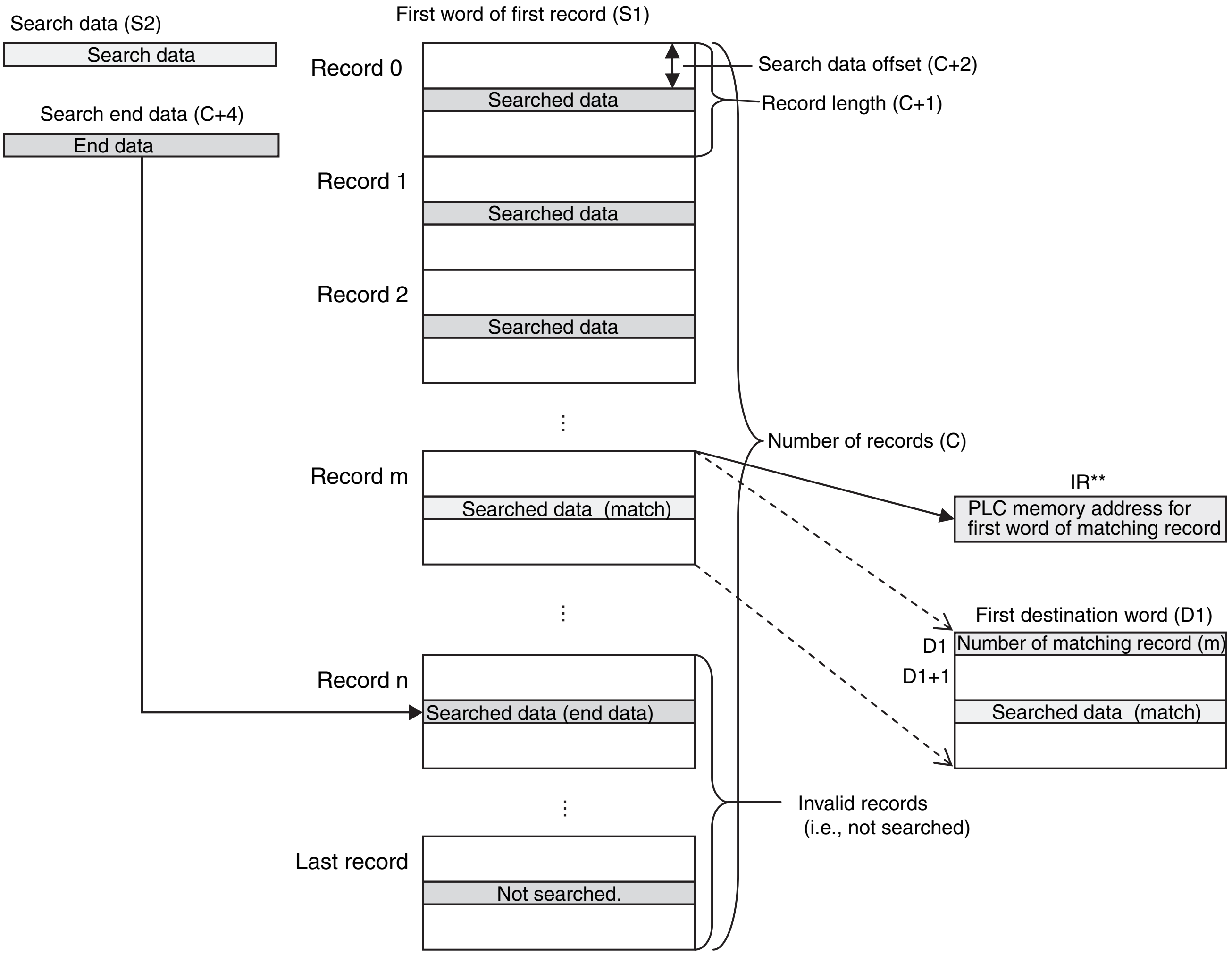
● Searching with the Setting to End the Search Disabled

The search is performed for the number of records specified in C.



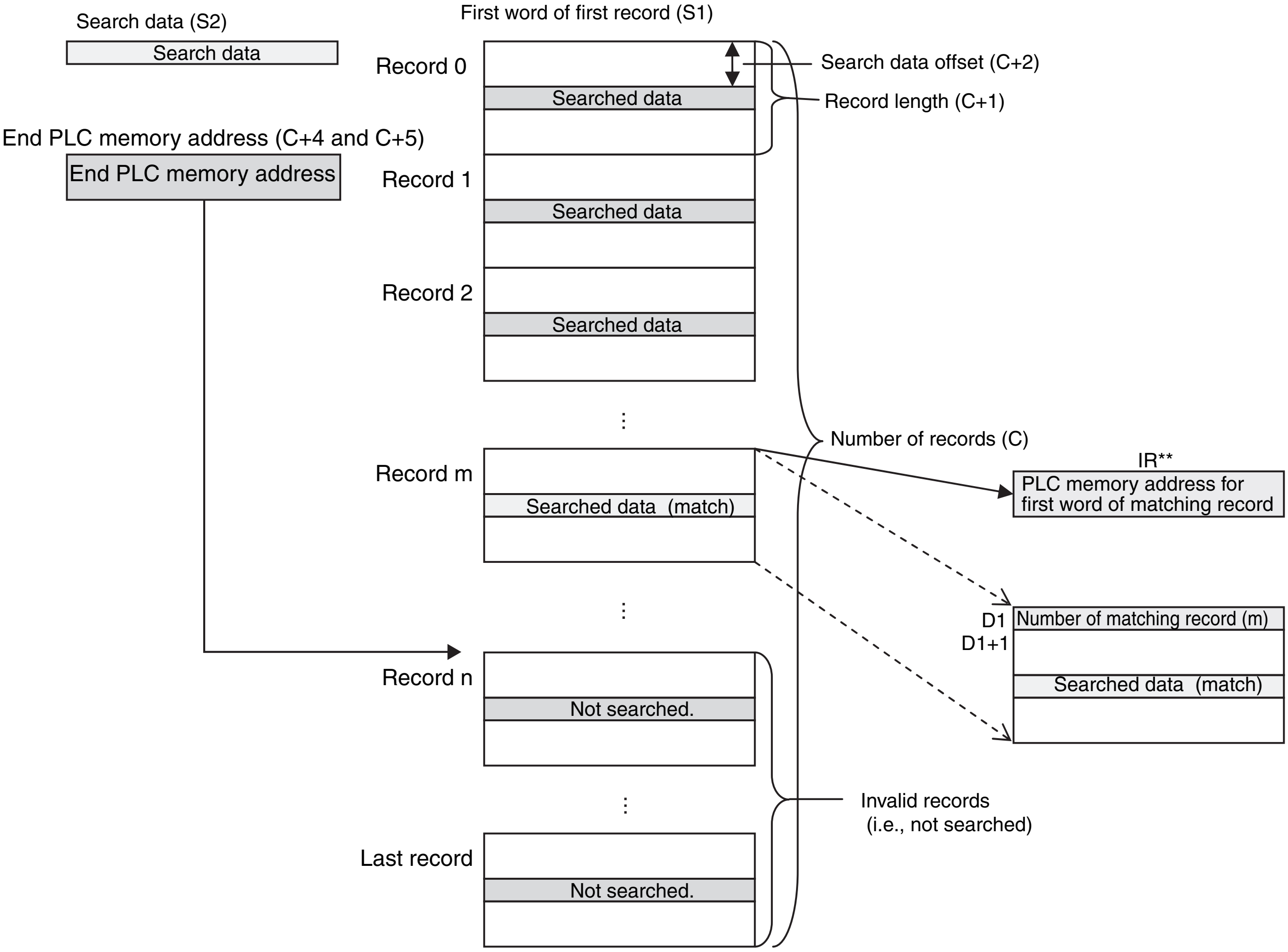
● Searching with End Data Enabled to End the Search

If the setting to end the search is enabled in C+3 for end data, records will be searched until the record just before the record in which the end data specified in C+4 is found. If the end data is not found, the number of records specified in C will be searched.



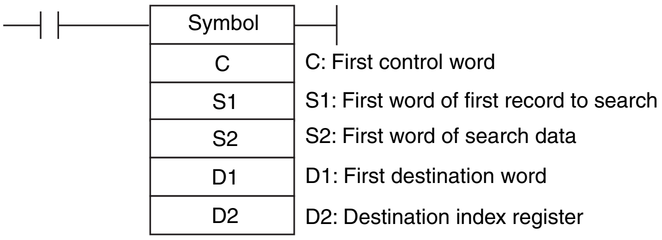
● Searching with End PLC Memory Address Enabled to End the Search

If the setting to end the search is enabled in C+3 for a PLC memory address, records will be searched until the PLC memory address specified in C+4 and C+5. No records past the end PLC memory address will be searched. If the end PLC memory address is not found, the number of records specified in C will be searched.



RSRCH2<, RSRCH2<=, RSRCH2=, RSRCH2>, RSRCH2>=

Instruction	Mnemonic	Variations	Function code	Function
Unsigned Two-word Record Search Instructions	RSRCH2< RSRCH2<= RSRCH2= RSRCH2> RSRCH2>=	@RSRCH2< @RSRCH2<= @RSRCH2= @RSRCH2> @RSRCH2>=	370 371 372 373 374	An Unsigned Two-word Record Search Instruction searches the specified table of records for a record with the specific data (2 words) in the specified location. When a record matching the specified condition is found, its record number and data are output.

	RSRCH2<, RSRCH2<=, RSRCH2=, RSRCH2>, RSRCH2>=
Symbol	

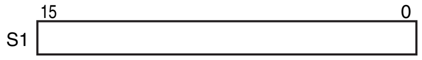
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

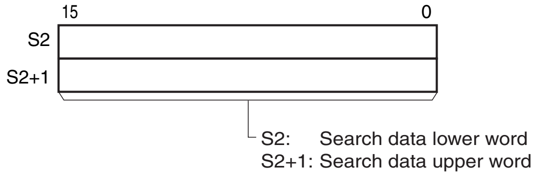
Operands

Operand	Description	Data type	Size
C	First control word	WORD	6
S1	First word of first record to search	WORD	Variable
S2	First word of search data	DWORD	2
D1	First destination word	WORD	Variable
D2	Destination index register	DWORD	2

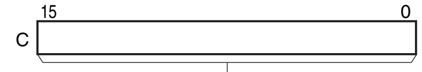
S1: First word of first record to search



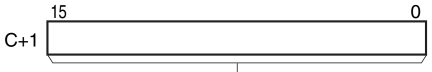
S2: First search data word



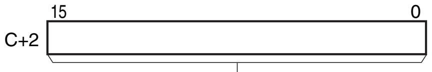
C: First control word



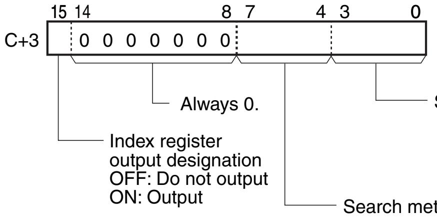
Number of records: 1 to 16384 decimal (maximum: 16,384)



Record length (in words): 2 to 1024 decimal (maximum: 1,024)



Search data offset (in words): 0 to 1022 decimal

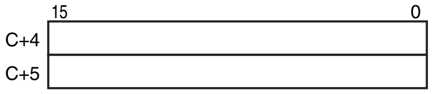


Always 0.

Index register output designation
OFF: Do not output
ON: Output

Setting to end search
0 hex: Disabled
1 hex: Enabled for end data (only for linear searches)
2 hex: Enabled for end PLC memory address

Search method
0 hex: Linear (Searches records in order from the specified first record.)
1 hex: Split search, ascending
2 hex: Split search, descending

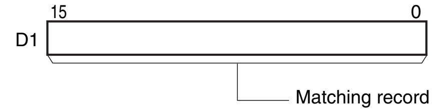


Setting to End Search Disabled
C+4: Always 0.
C+5: Always 0.

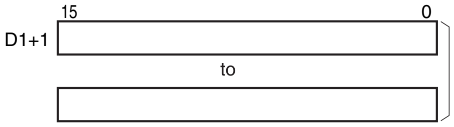
End Data Enabled
C+4 and C+5: End data (0000 0000 to FFFF FFFF hex)

End PLC Memory Address Enabled
C+4: Lower word of PLC memory address
C+5: Upper word of PLC memory address

D1: First destination word

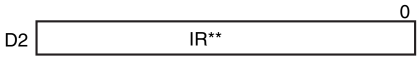


Matching record number
Contains the record number of the matching record.
(The first record is record 0.)



Matching record data
(All record data for record length.)

D2: Destination index register



The PLC memory address of the first word of the matching record is stored in the specified index register. If index register output is not specified in C+3, then 0000 0000 hex is stored in D2.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S1																		
S2											OK							
D1																		
D2	---	---	---	---	---	---	---	---	---	---			OK	---				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON when the record length in C+1 is not between 2 and 1,024, inclusive. • ON when the number of records specified in C times the record length in C+1 is greater than 32,768. • ON when the number of records specified in C times the record length in C+1 is 0. • ON when the search data offset in C+2 is greater than two less than the record length in C+1. • OFF in all other cases.
Equals Flag	P_EQ	• ON when the search data is found. • OFF when the search data is not found.
Carry Flag	P_CY	Linear Search • ON when a matching record is found before the final record and before the condition for the setting to end the search is met. • OFF when the search is completed through the final record or until the condition for the setting to end the search is met. Split Search • Always OFF.

Function

The number of records specified by C starting from the record specified by S1 are searched for the data specified by S2 and S2+1. If a record that matches the search conditions is found, the Equals Flag will turn ON, the record number of the matching record will be output to D1 and the matching record data will be output starting from D1+1. If outputting to an index register is specified in C+3, the PLC memory address of the first word of the matching record is output to the specified index register. When this happens, the Equals Flag is turned ON. If more than one matching record is found, the record closest to the first record that is searched will be output.

If a matching record is not found, nothing will be output starting from D1 (the previous values will be maintained).

The search is performed by the method specified in C+3. There are two search methods.

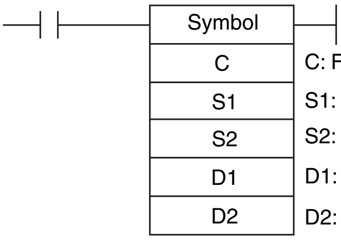
- Linear Search
  For a linear search, the records are searched in order from the specified first record. Linear searching is used when the search data has not been sorted. If a matching record is found before the maximum number of records specified in C is searched, the Carry Flag will be turned ON. (This is to indicate that there are records that were not searched.)
- Split Search (Ascending or Descending)
  If the records that are being searched have been sorted in ascending or descending order, a split search can be used to reduce the instruction execution time.

Precautions

- If the records being searched have not been sorted in ascending or descending order when a split search is specified, the results of the search will not be accurate.
- A split search cannot be specified if using search end data has been enabled. If using search end data is enabled, a linear search will be performed even if a split search is specified.

RSRCH4<, RSRCH4<=, RSRCH4=, RSRCH4>, RSRCH4>=

Instruction	Mnemonic	Variations	Function code	Function
Unsigned Four-word Record Search Instructions	RSRCH4< RSRCH4<= RSRCH4= RSRCH4> RSRCH4>=	@RSRCH4< @RSRCH4<= @RSRCH4= @RSRCH4> @RSRCH4>=	380 381 382 383 384	An Unsigned Four-word Record Search Instruction searches the specified table of records for a record with the specific data (4 words) in the specified location. When a record matching the specified condition is found, its record number and data are output.

	RSRCH4<, RSRCH4<=, RSRCH4=, RSRCH4>, RSRCH4>=
Symbol	 C: First control word S1: First word of first record to search S2: First word of search data D1: First destination word D2: Destination index register

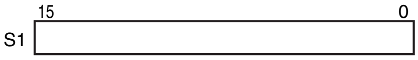
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	First control word	WORD	8
S1	First word of first record to search	WORD	Variable
S2	First word of search data	LWORD	4
D1	First destination word	WORD	Variable
D2	Destination index register	DWORD	2

S1: First word of first record to search



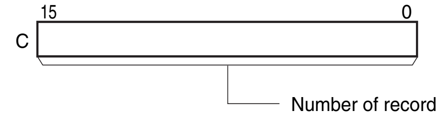
S2: First search data word

S2: Search data lowest word
S2+1: Search data lower word
S2+2: Search data upper word
S2+3: Search data highest word

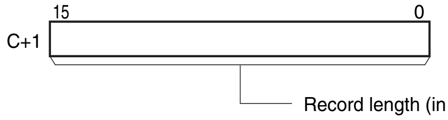
The data would be set as follows to search for #0123456789ABCDEF.

S2	#CDEF
S2+1	#89AB
S2+2	#4567
S2+3	#0123

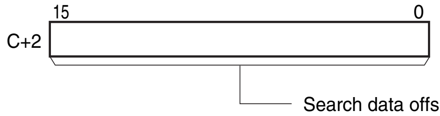
C: First control word



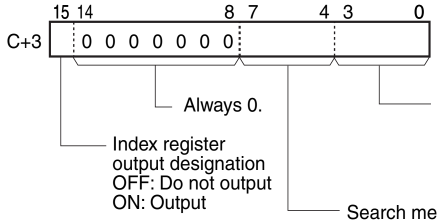
Number of records:
1 to 8192 decimal (maximum: 8,192)



Record length (in words):
4 to 1024 decimal (maximum: 1,024)



Search data offset (in words):
0 to 1010 decimal



Always 0.

Index register
output designation
OFF: Do not output
ON: Output

Setting to end search
0 hex: Disabled
1 hex: Enabled for end data (only for linear searches)
2 hex: Enabled for end PLC memory address

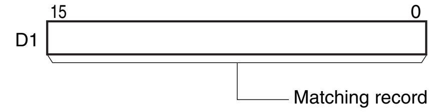
Search method
0 hex: Linear (Searches records in order from the specified first record.)
1 hex: Split search, ascending
2 hex: Split search, descending

Setting to End Search Disabled
C+4: Always 0.
C+5: Always 0.
C+6: Always 0.
C+7: Always 0.

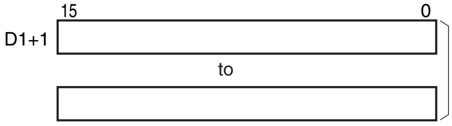
End Data Enabled
C+4 to C+7: End data
(0000 0000 0000 0000 to FFFF FFFF FFFF FFFF hex)
Store the end data in the following order starting from C+4:
Lowest 16 bits, lower 16 bits, higher 16 bits, highest 16 bits.

End PLC Memory Address Enabled
C+4: Lower word of PLC memory address
C+5: Upper word of PLC memory address
C+6: Always 0.
C+7: Always 0.

D1: First destination word



Matching record number
Contains the record number of the matching record.
(The first record is record 0.)



Matching record data
(All record data for record length.)

D2: Destination index register

The PLC memory address of the first word of the matching record is stored in the specified index register. If index register output is not specified in C+3, then 0000 0000 hex is stored in D2.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C S1 S2 D1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D2	---	---	---	---	---	---	---	---	---	---	OK		OK	---				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON when the record length in C+1 is not between 4 and 1,024, inclusive. • ON when the number of records specified in C times the record length in C+1 is greater than 32,768. • ON when the number of records specified in C times the record length in C+1 is 0. • ON when the search data offset in C+2 is greater than four less than the record length in C+1. • OFF in all other cases.
Equals Flag	P_EQ	• ON when the search data is found. • OFF when the search data is not found.
Carry Flag	P_CY	Linear Search • ON when a matching record is found before the final record and before the condition for the setting to end the search is met. • OFF when the search is completed through the final record or until the condition for the setting to end the search is met. Split Search • Always OFF.

Function

The number of records specified by C starting from the record specified by S1 are searched for the data specified by S2 to S2+3. If a record that matches the search conditions is found, the Equals Flag will turn ON, the record number of the matching record will be output to D1 and the matching record data will be output starting from D1+1. If outputting to an index register is specified in C+3, the PLC memory address of the first word of the matching record is output to the specified index register. When this happens, the Equals Flag is turned ON. If more than one matching record is found, the record closest to the first record that is searched will be output. *

If a matching record is not found, nothing will be output starting from D1 (the previous values will be maintained).

The search is performed by the method specified in C+3. There are two search methods.

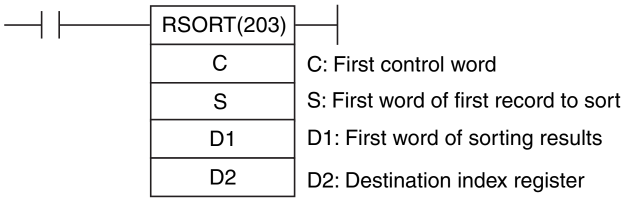
- Linear Search
  For a linear search, the records are searched in order from the specified first record. Linear searching is used when the search data has not been sorted. If a matching record is found before the maximum number of records specified in C is searched, the Carry Flag will be turned ON. (This is to indicate that there are records that were not searched.)
- Split Search (Ascending or Descending)
  If the records that are being searched have been sorted in ascending or descending order, a split search can be used to reduce the instruction execution time.

Precautions

- If the records being searched have not been sorted in ascending or descending order when a split search is specified, the results of the search will not be accurate.
- A split search cannot be specified if using search end data has been enabled. If using search end data is enabled, a linear search will be performed even if a split search is specified.

RSORT

Instruction	Mnemonic	Variations	Function code	Function
UNSIGNED ONE-WORD RECORD SORT	RSORT	@RSORT	203	Sorts the records in the specified table according to the data (1 word) at the specified position in the records.

	RSORT
Symbol	

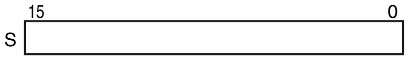
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

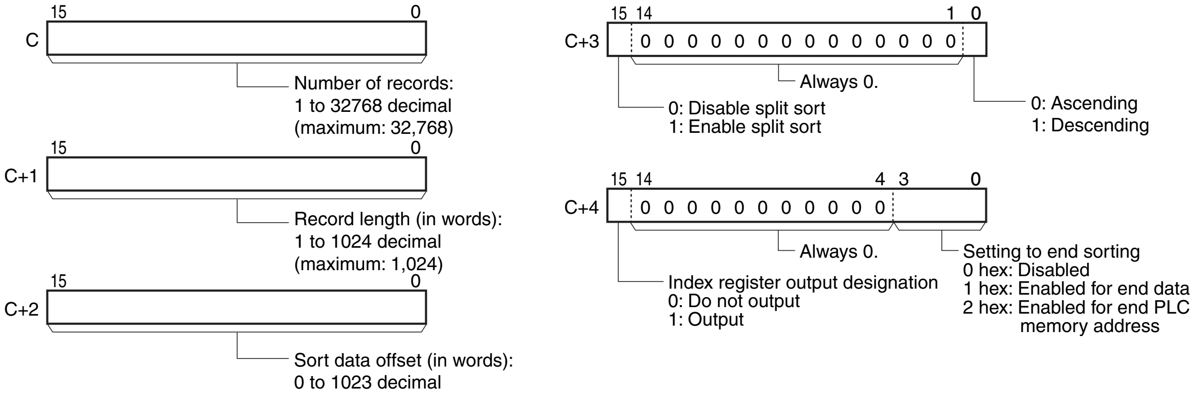
Operands

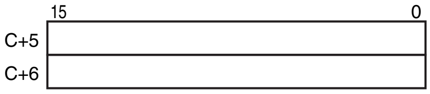
Operand	Description	Data type	Size
C	First control word	WORD	87
S1	First word of first record to sort	WORD	Variable
D1	First word of sorting results	WORD	1
D2	Destination index register	DWORD	2

S: First word of first record to sort



C: First control word

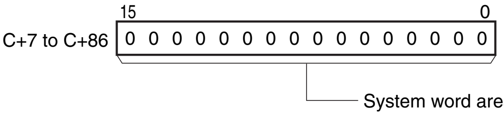




Searching with the Setting to End the Sort Disabled
C+5: Always 0.
C+6: Always 0.

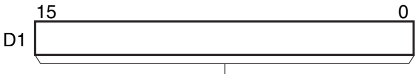
Searching with End Data Enabled to End the Sort
C+5: Sort end data (0000 to FFFF hex)
C+6: Always 0.

Searching with End Data Enabled to End the Sort
C+5: Lower word of end PLC memory address
C+6: Upper word of end PLC memory address



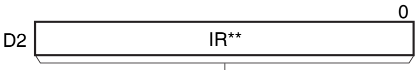
System word area
Always clear C+7 to 0 before starting to sort records.

D1: First word of sorting results



Number of records that were sorted
This word contains the number of records that were sorted.

D2: Destination index register



One higher than the PLC memory address of the last word in the range of records that was sorted. If index register output is not specified in D+4, then 0000 0000 hex is stored in D2.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S																		
D1												OK						
D2	---	---	---	---	---	---	---	---	---	---	OK	---	OK	---				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON when the record length in C+1 is not between 1 and 1,024, inclusive. • ON when the number of records specified in C times the record length in C+1 is greater than 32,768. • ON when the number of records specified in C times the record length in C+1 is 0. • ON when the sort data offset in C+2 is greater than one less than the record length in C+1. • OFF in all other cases.
Carry Flag	P_CY	• ON while the records are being sorted. • OFF when the sort operation has been completed.

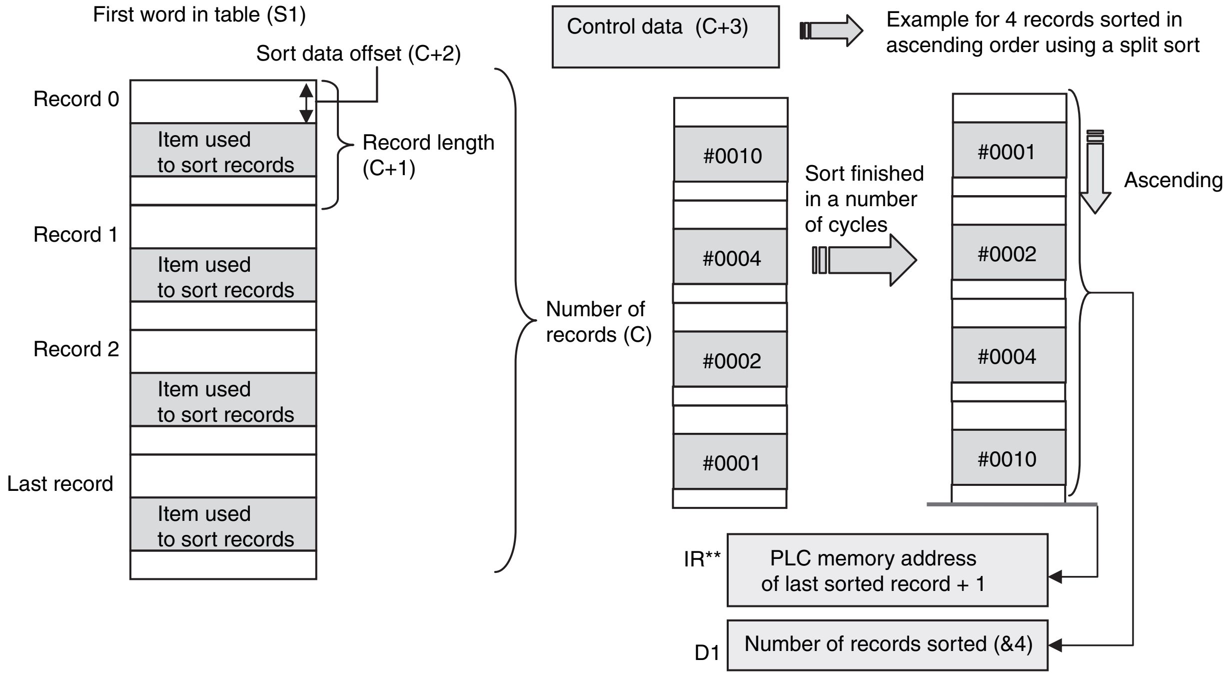
Function

The number of records specified by C starting from the record specified by S are sorted according to the data in the specified location (1 word). When the sort operation has been completed, the Carry Flag will turn OFF and the number of records that were sorted will be output to D1. If outputting to an index register is specified in C+4, one address higher than the PLC memory address of the last word in the sorted records is output to the specified index register. The sort direction is specified as ascending or descending in C+3.

If split sorting is enabled in C+3, the sort will be executed over multiple cycles. The Carry Flag will be ON while the sort operation is being executed and will turn OFF when the sort operation has been completed. If the execution condition of RSORT turns OFF before the sort operation has been finished, the sort will be continued the next time the execution condition turns ON.

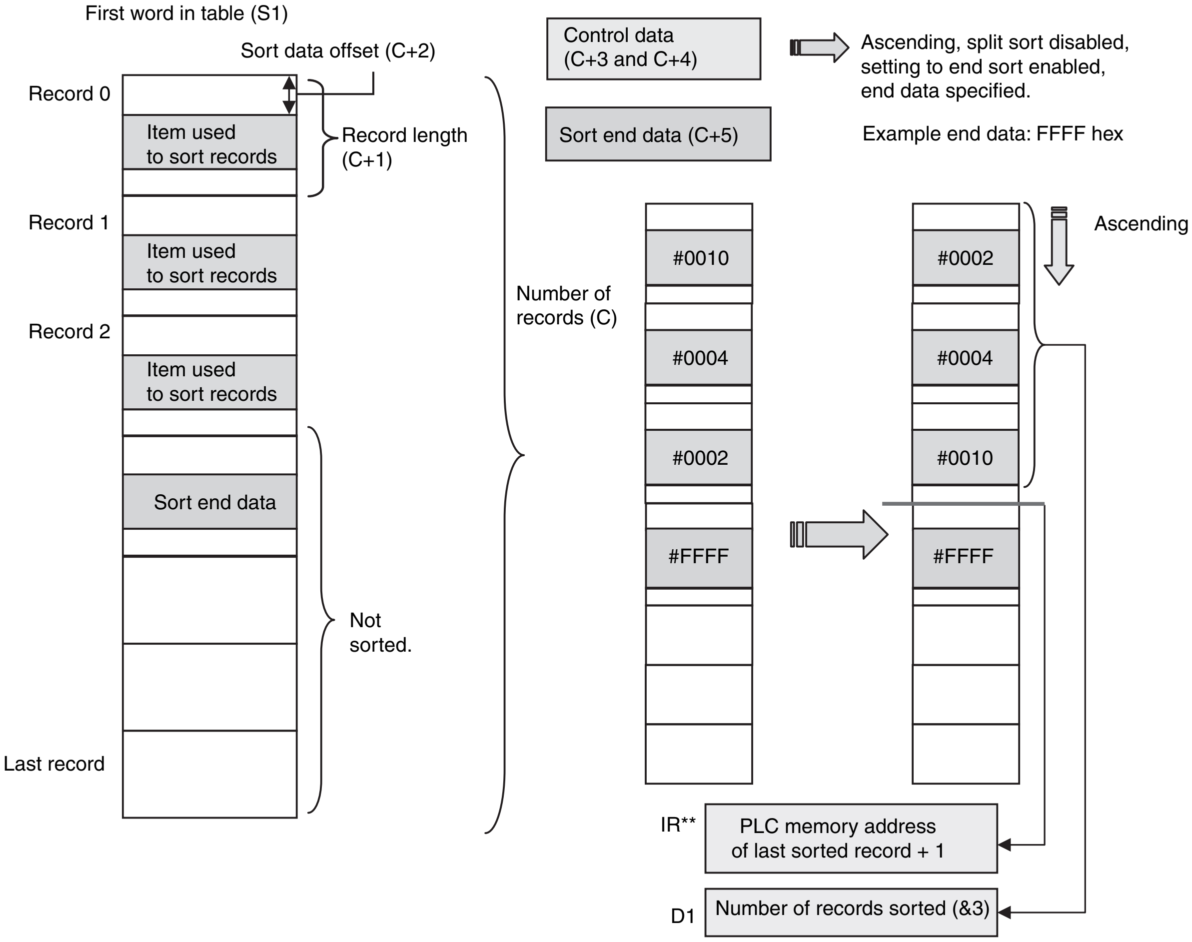
● Sorting with the Setting to End Sorting Disabled

The sort operation is performed for the number of records specified in C.



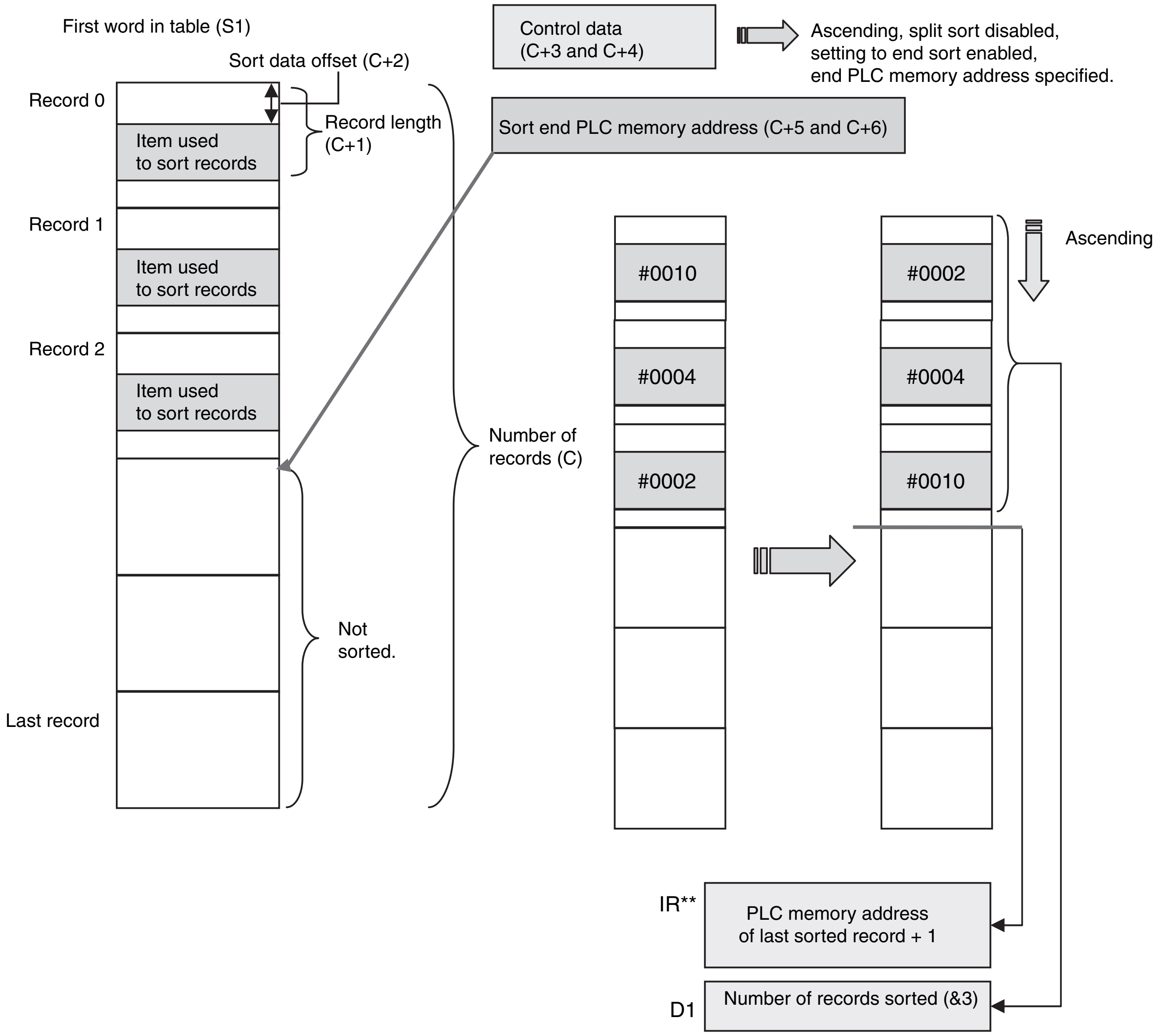
● Sorting with End Data Enabled to End the Sort

If the setting to end the sort is enabled in C+4 for end data, records will be sorted through the record just before the record in which the end data specified in C+5 is found. If the end data is not found, the number of records specified in C will be sorted.



● Searching with End PLC Memory Address Enabled to End the Search

If the setting to end the search is enabled in C+3 for a PLC memory address, records will be searched until the PLC memory address specified in C+4 and C+5. No records past the end PLC memory address will be searched. If the end PLC memory address is not found, the number of records specified in C will be searched.

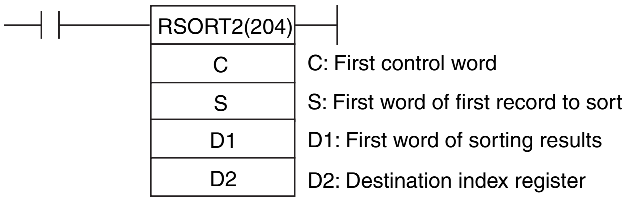


Precautions

Always clear C+7 to 0 before starting to sort.

RSORT2

Instruction	Mnemonic	Variations	Function code	Function
UNSIGNED TWO-WORD RECORD SORT	RSORT2	@RSORT2	204	Sorts the records in the specified table according to the data (2 words) at the specified position in the records.

	RSORT2
Symbol	

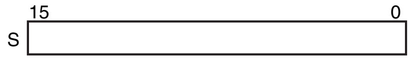
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

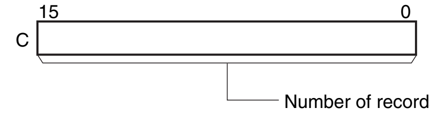
Operands

Operand	Description	Data type	Size
C	First control word	WORD	87
S1	First word of first record to sort	WORD	Variable
D1	First word of sorting results	WORD	1
D2	Destination index register	DWORD	2

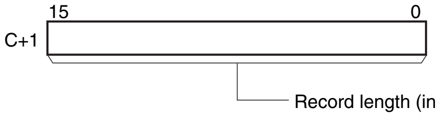
S: First word of first record to sort



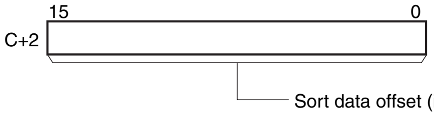
C: First control word



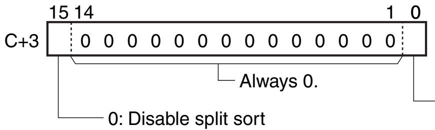
Number of records: 1 to 16384 decimal (maximum: 16,384)



Record length (in words): 2 to 1024 decimal (maximum: 1,024)



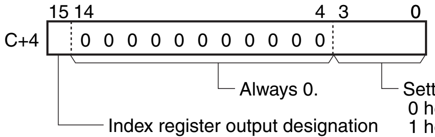
Sort data offset (in words): 0 to 1022 decimal



Always 0.

0: Disable split sort
1: Enable split sort

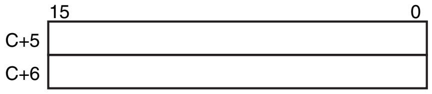
0: Ascending
1: Descending



Always 0.

Index register output designation
0: Do not output
1: Output

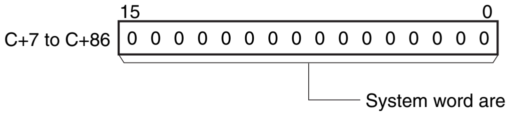
Setting to end sorting
0 hex: Disabled
1 hex: Enabled for end data
2 hex: Enabled for end PLC memory address



Searching with the Setting to End the Sort Disabled
C+5: Always 0.
C+6: Always 0.

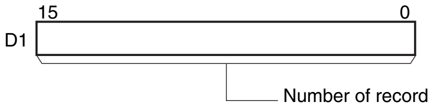
Searching with End Data Enabled to End the Sort
C+5: Sort end data
(0000 0000 to FFFF FFFF hex)
C+6: Sort end data (0000 0000 to FFFF FFFF hex)

Searching with PLC Memory Address Enabled to End the Sort
C+5: Lower word of end PLC memory address
C+6: Upper word of end PLC memory address



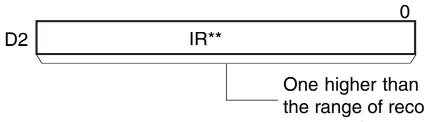
System word area
Always clear C+7 to 0 before starting to sort records.

D1: First word of sorting results



Number of records that were sorted
This word contains the number of records that were sorted.

D2: Destination index register



One higher than the PLC memory address of the last word in the range of records that was sorted. If index register output is not specified in D+4, then 0000 0000 hex is stored in D2.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S																		
D1												OK						
D2	---	---	---	---	---	---	---	---	---	---	OK	---	OK	---				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON when the record length in C+1 is not between 2 and 1,024, inclusive. • ON when the number of records specified in C times the record length in C+1 is greater than 32,768. • ON when the number of records specified in C times the record length in C+1 is 0. • ON when the sort data offset in C+2 is greater than two less than the record length in C+1. • OFF in all other cases.
Carry Flag	P_CY	• ON while the records are being sorted. • OFF when the sort operation has been completed.

Function

The number of records specified by C starting from the record specified by S are sorted according to the data in the specified location (2 words). When the sort operation has been completed, the Carry Flag will turn OFF and the number of records that were sorted will be output to D1. If outputting to an index register is specified in C+4, one address higher than the PLC memory address of the last word in the sorted records is output to the specified index register. The sort direction is specified as ascending or descending in C+3.

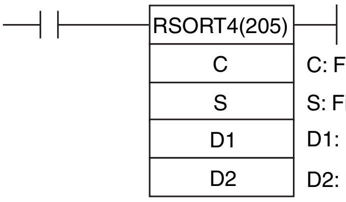
If split sorting is enabled in C+3, the sort will be executed over multiple cycles. The Carry Flag will be ON while the sort operation is being executed and will turn OFF when the sort operation has been completed. If the execution condition of RSORT2 turns OFF before the sort operation has been finished, the sort will be continued the next time the execution condition turns ON.

Precautions

Always clear C+7 to 0 before starting to sort.

RSORT4

Instruction	Mnemonic	Variations	Function code	Function
UNSIGNED FOUR-WORD RECORD SORT	RSORT4	@RSORT4	205	Sorts the records in the specified table according to the data (4 words) at the specified position in the records.

	RSORT4
Symbol	 C: First control word S: First word of first record to sort D1: First word of sorting results D2: Destination index register

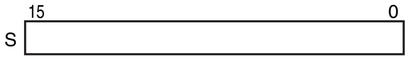
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

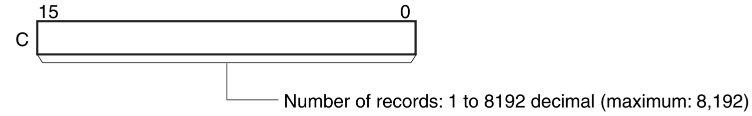
Operands

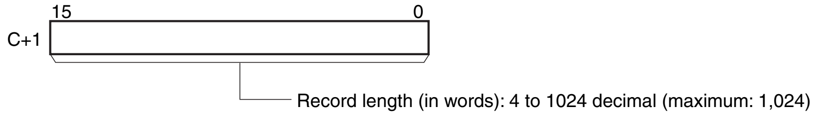
Operand	Description	Data type	Size
C	First control word	WORD	89
S1	First word of first record to sort	WORD	Variable
D1	First word of sorting results	WORD	1
D2	Destination index register	DWORD	2

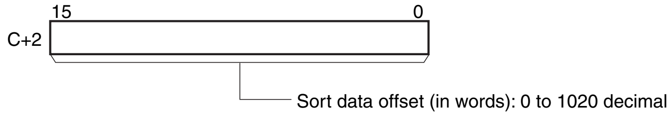
S: First word of first record to sort

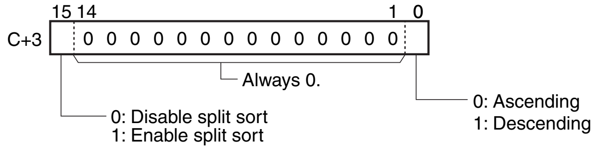


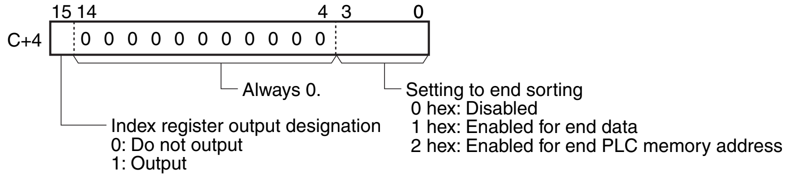
C: First control word







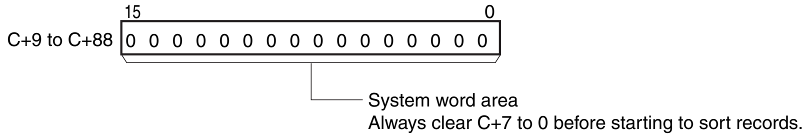




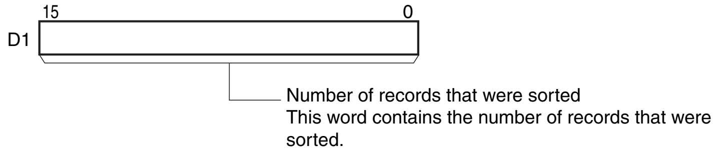
Setting to End Sorting Disabled
C+5: Always 0.
C+6: Always 0.

End Data Enabled
C+5 to C+8: Sort end data
(0000 0000 0000 0000 to FFFF FFFF FFFF FFFF hex)
Store the end data in the following order starting from C+5: Lowest 16 bits, lower 16 bits, higher 16 bits, highest 16 bits.

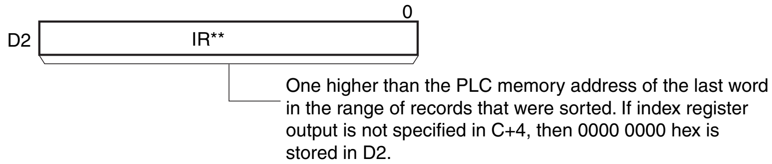
End PLC Memory Address Enabled
C+5: Lower word of end PLC memory address
C+6: Upper word of end PLC memory address
C+7: Always 0.
C+8: Always 0.



D1: First word of sorting results



D2: Destination index register



Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S																		
D1												OK						
D2	---	---	---	---	---	---	---	---	---	---	OK	---	OK	---				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON when the record length in C+1 is not between 4 and 1,024, inclusive. • ON when the number of records specified in C times the record length in C+1 is greater than 32,768. • ON when the number of records specified in C times the record length in C+1 is 0. • ON when the sort data offset in C+2 is greater than four less than the record length in C+1. • OFF in all other cases.
Carry Flag	P_CY	• ON while the records are being sorted. • OFF when the sort operation has been completed.

Function

The number of records specified by C starting from the record specified by S are sorted according to the data in the specified location (4 words). When the sort operation has been completed, the Carry Flag will turn OFF and the number of records that were sorted will be output to D1. If outputting to an index register is specified in C+4, one address higher than the PLC memory address of the last word in the sorted records is output to the specified index register. The sort direction is specified as ascending or descending in C+3.

If split sorting is enabled in C+3, the sort will be executed over multiple cycles. The Carry Flag will be ON while the sort operation is being executed and will turn OFF when the sort operation has been completed. If the execution condition of RSORT4 turns OFF before the sort operation has been finished, the sort will be continued the next time the execution condition turns ON.

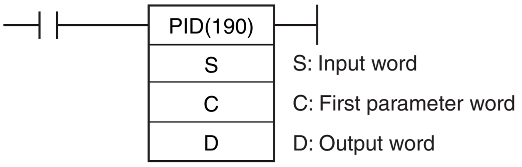
Precautions

Always clear C+7 to 0 before starting to sort.

Data Control Instructions

PID

Instruction	Mnemonic	Variations	Function code	Function
PID CONTROL	PID	---	190	Executes PID control according to the specified parameters.

	PID
Symbol	

Applicable Program Areas

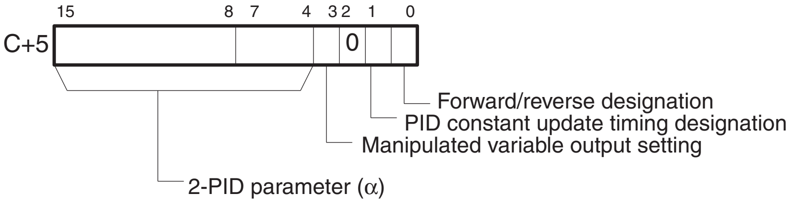
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

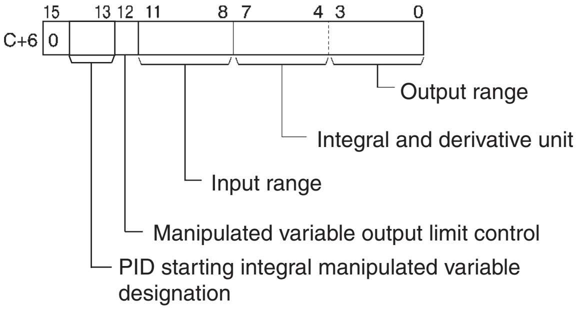
Operands

Operand	Description	Data type	Size
S	Input word	UINT	1
C	First parameter word	WORD	39
D	Output word	UINT	1

C: First Parameter Word

	15 … 0
C	Set value (SV)
C+1	Proportional band (P)
C+2	Integral constant (Tik)
C+3	Derivative constant (Tdk)
C+4	Sampling period (τ)





	15 … 0
C+7	Manipulated variable lower limit
C+8	Manipulated variable upper limit
C+9 C+38	Work area (30 words: Cannot be used by user.)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
C												---						
D												OK						

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.

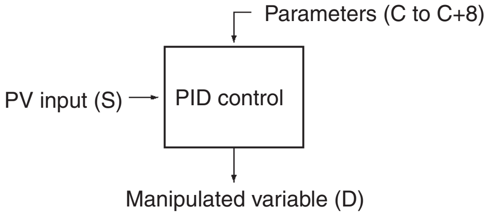
Function

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.

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.)

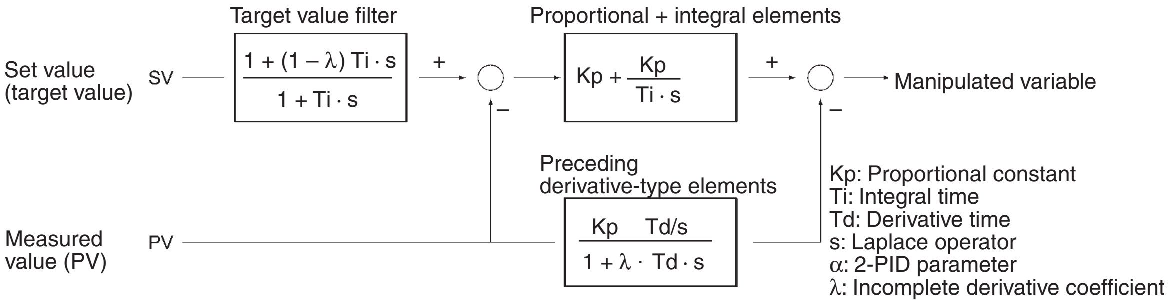
Calculation Method

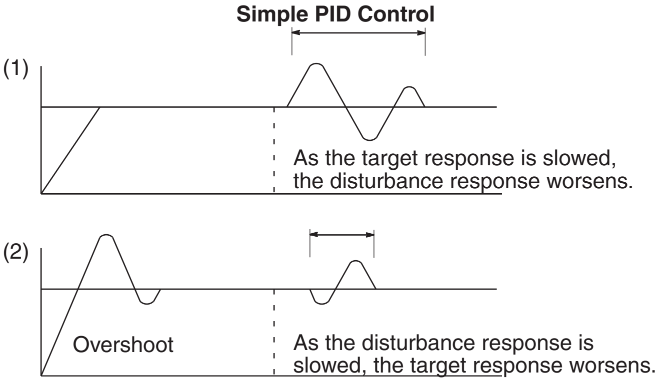
Calculations in PID control are performed by the target value filtered control with two degrees of freedom.

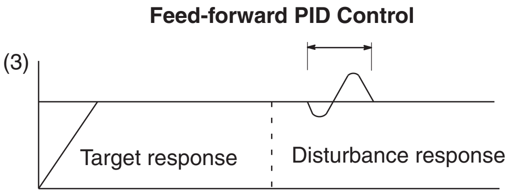
Block Diagram for Target Value PID with Two Degrees of Freedom

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) When “1” is specified for the integral/differential constant: 1 to 8191 When “9” is specified: 0.1 to 819.1 s (See note 1.)	
C+3	Tdk Derivative Constant	A constant expressing the strength of the derivative action. As this value increases, the derivative strength increases.	0001 to 1FFF hex (1 to 8191); (0000 = Derivative operation not executed) When “1” is specified for the integral/differential constant: 1 to 8191 When “9” is specified: 0.1 to 819.1 s (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 when the PV equals the SV. This setting is enabled when there is no integral operation.	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
Bit 00 of C+5	PID forward/reverse designation	Determines the direction of the proportional action.	0: Reverse action 1: Forward action	Not allowed
Bits 13 to 14 of C+6	ID starting integral manipulated variable designation (See note 4.)	Determines the initial integral manipulated variable when PID control is started (i.e., when the input turns ON).	Bit 14 = 0 and bit 13=0: Start from same integral manipulated value as manipulated variable output designation (Pre-Ver. 4.0 operation). Bit 14 = 0 or 1 and bit 13 = 1: Bumpless operation (i.e., start from an integral manipulated variable that will not abruptly change the manipulated variable output and result in a continuous change). Bit 14 = 1 and bit 13 = 0: Start with integral manipulated variable = 0.	
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 1: 9 bits 2: 10 bits 3: 11 bits 4: 12 bits 5: 13 bits 6: 14 bits 7: 15 bits 8: 16 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) (See note 1.)	
Bits 00 to 03 of C+6	Output range	The number of output data bits.	0: 8 bits 1: 9 bits 2: 10 bits 3: 11 bits 4: 12 bits 5: 13 bits 6: 14 bits 7: 15 bits 8: 16 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 ≤ MV output lower limit ≤ MV output upper limit ≤ Max. value of output range

4 CS1-H CPU Units, CJ1-H CPU Units, CJ1M CPU Units, and CS1D CPU Units (for Single CPU Systems) with unit version 4.0 or later, and CJ2 CPU Units only.

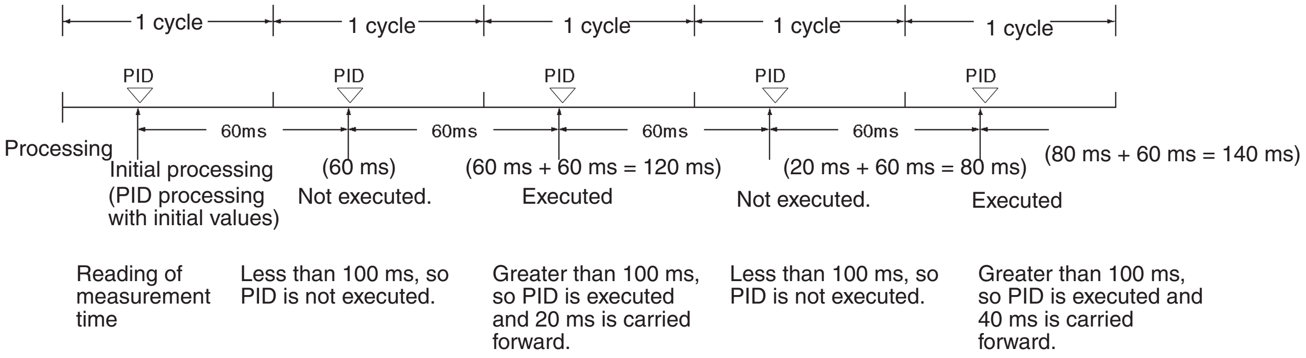
Sampling Period and Cycle Time

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.



PID control

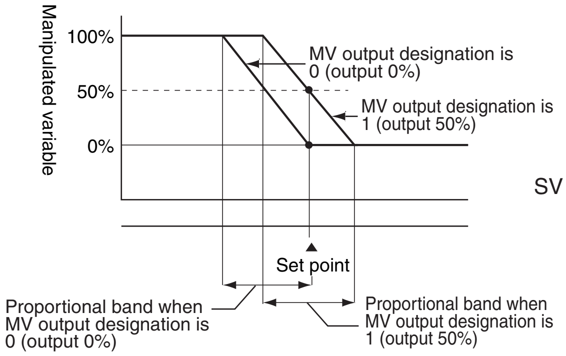
● Proportional Action (P)

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.

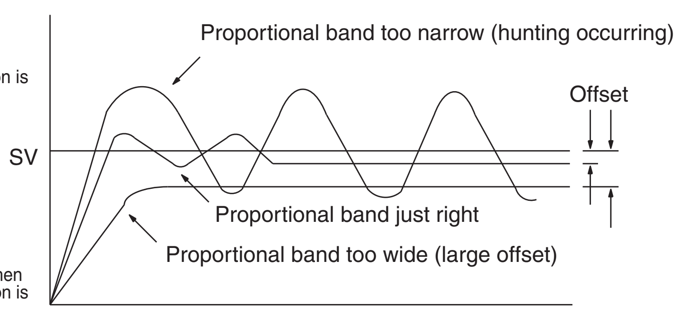
If the proportional action is used and the present value (PV) becomes smaller than the proportional band, the manipulated variable (MV) is 100% (i.e., the maximum value). Within the proportional band, the MV is made proportional to the deviation (the difference between from SV and PV) and gradually decreased 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)



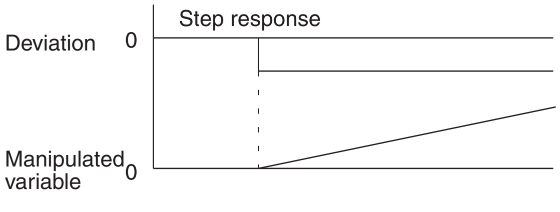
Adjusting the Proportional Band



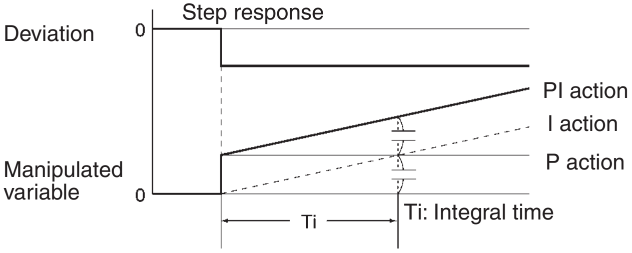
● Integral Action (I)

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.

Integral Action



Pi Action and Integral Time

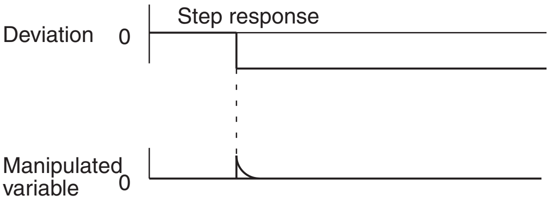


● Derivative Action (D)

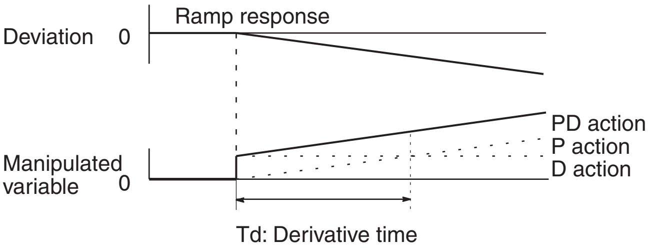
Proportional action and integral action both make corrections with respect to the control results, so there is inevitably a response delay. Derivative action compensates for that drawback. In response to a sudden disturbance it delivers a large manipulated variable and rapidly restores the original status. 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



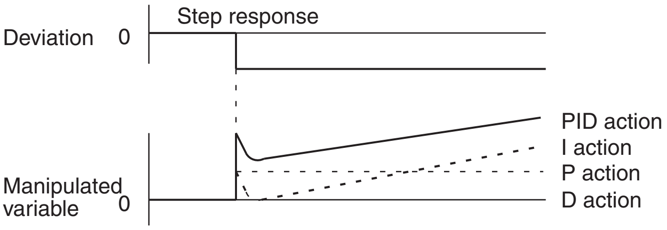
PD Action and Derivative Time



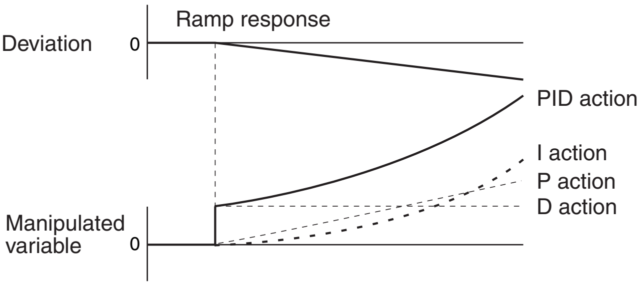
PID Action

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



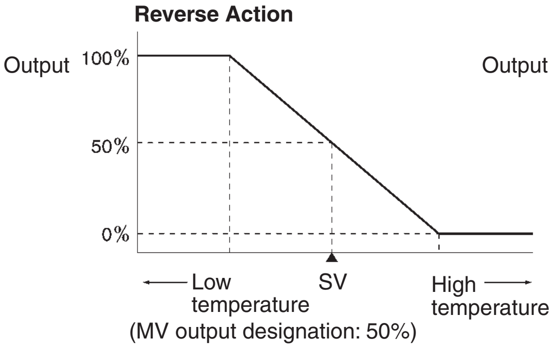
Ramp Response of PID Control Action Output

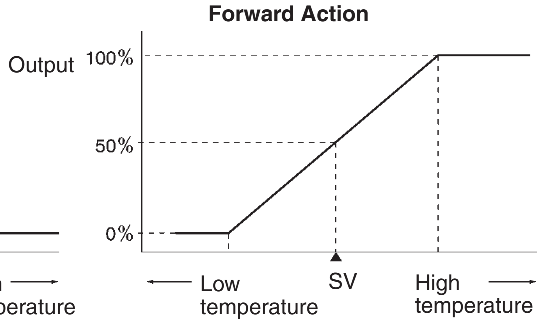


Direction of Action

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.

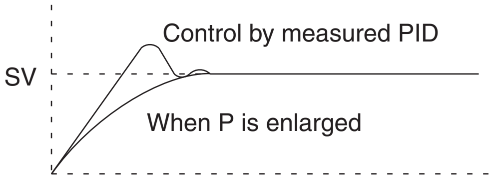




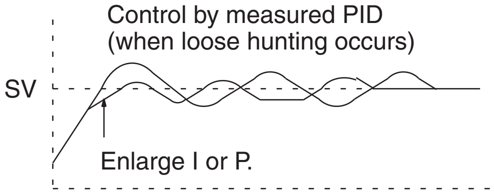
Adjusting PID Parameters

The general relationship between PID parameters and control status is shown below.

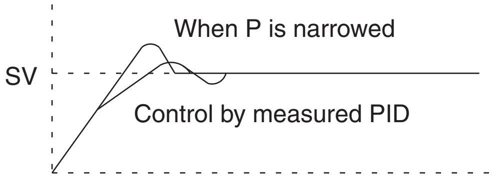
- When it is not a problem if a certain amount of time is required for stabilization (settlement time), but it is important not to cause overshooting, then enlarge the proportional band.



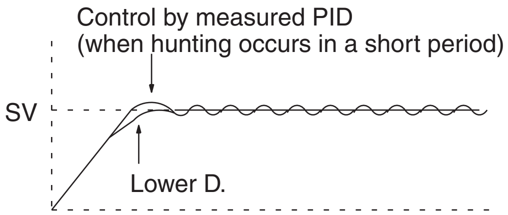
- When there is broad hunting, or when operation is tied up by overshooting and undershooting, it is probably because integral action is too strong. The hunting will be reduced if the integral time is increased or the proportional band is enlarged.



- 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.



- 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.



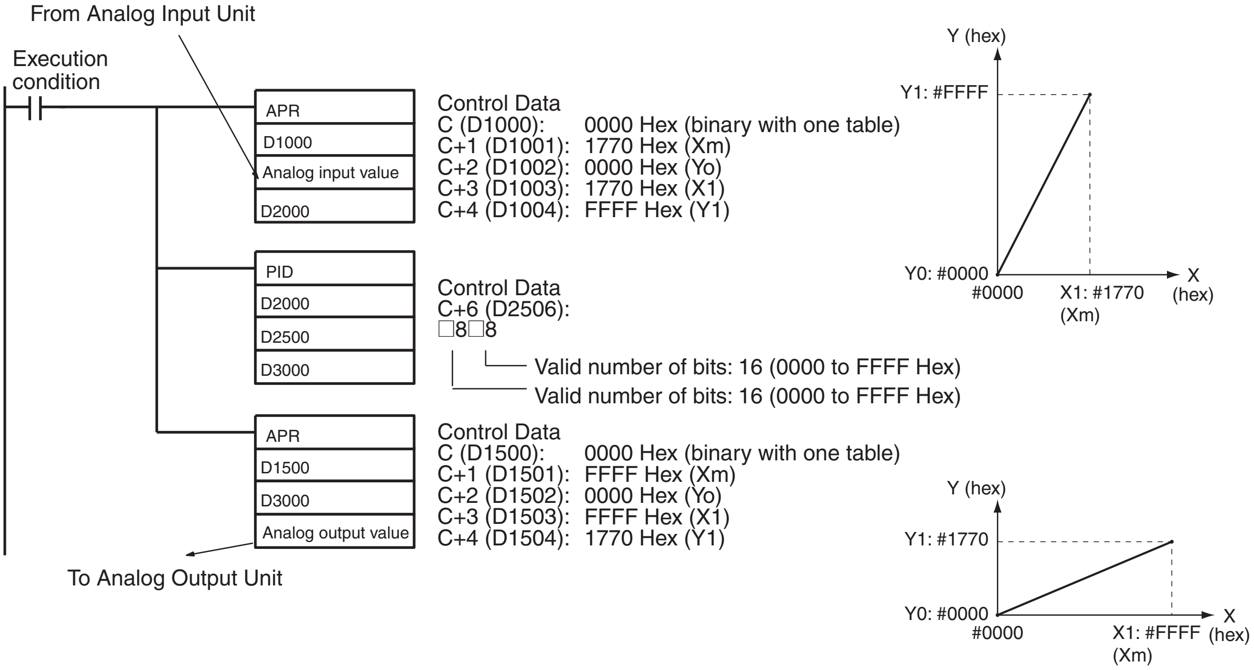
Hint

- The setting in bit 1 of C+5 is supported only by CJ2, CJ1, CS1-H, CJ1-H CPU Units and CS1 CPU Units with lot numbers of 001201□□□□ or later (manufactured December 1, 2000 or later).
- 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).

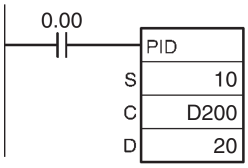
The following program section shows an example for a DRT1-AD04 Analog Input Unit and DRT1-DA02 Analog Output Unit operating as DeviceNet slaves. The data ranges for these two Units is 0000 to 1770 hex, which cannot be specified merely by setting the valid number of digits. APR(069) is thus used to convert 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.



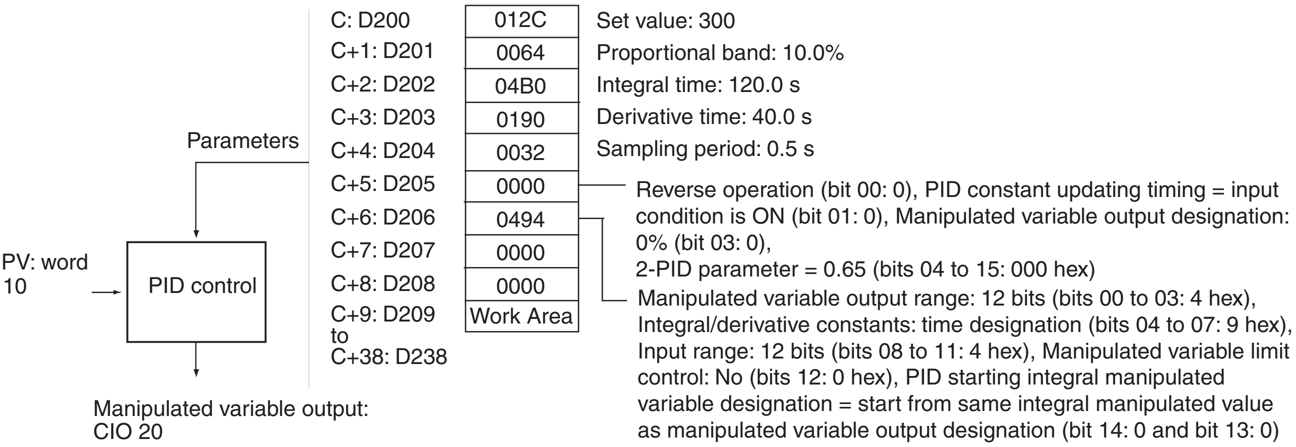
Precautions

- If PID(190) is used between IL(002) and ILC(003), between JMP(004) and JME(005), or in a subroutine, operation will not be consistent depending on the input condition.
- 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.
- 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), therefore, when using the Always ON Flag (ON) as an input condition for a PID command, provide a separate process that clears C+9 to C+38 to 0 (writes zeros) when operation is started.
- A PID parameter storage word cannot be shared by multiple PID instructions. Even when the same parameter is used in multiple PID instructions, separate words must be specified.

Example Programming



- 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 20.
- 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 20.
- 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.

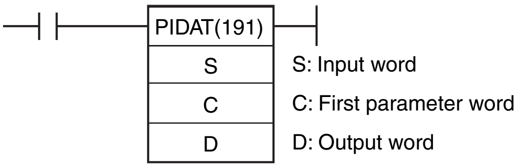


C: D200	012C	Set value: 300
C+1: D201	0064	Proportional band: 10.0%
C+2: D202	04B0	Integral time: 120.0 s
C+3: D203	0190	Derivative time: 40.0 s
C+4: D204	0032	Sampling period: 0.5 s
C+5: D205	0000	Reverse operation (bit 00: 0), PID constant updating timing = input condition is ON (bit 01: 0), Manipulated variable output designation: 0% (bit 03: 0), 2-PID parameter = 0.65 (bits 04 to 15: 000 hex)
C+6: D206	0494	Manipulated variable output range: 12 bits (bits 00 to 03: 4 hex), Integral/derivative constants: time designation (bits 04 to 07: 9 hex), Input range: 12 bits (bits 08 to 11: 4 hex), Manipulated variable limit control: No (bits 12: 0 hex), PID starting integral manipulated variable designation = start from same integral manipulated value as manipulated variable output designation (bit 14: 0 and bit 13: 0)
C+7: D207	0000	
C+8: D208	0000	
C+9: D209 to C+38: D238	Work Area	

Note When CIO 0.00 is OFF, operation can be the same as manual operation by writing to CIO 20.
When changing from manual operation to automatic operation by executing PID(190), extreme changes in the manipulated value are restricted. (The manipulated variable after switching to automatic operation will start at the previous value of the integral manipulated variable.)

PIDAT

Instruction	Mnemonic	Variations	Function code	Function
PID CONTROL WITH AUTOTUNING	PIDAT	---	191	Executes PID control according to the specified parameters. The PID constants can be autotuned.

	PID
Symbol	

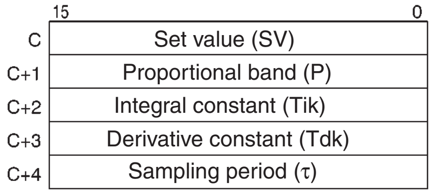
Applicable Program Areas

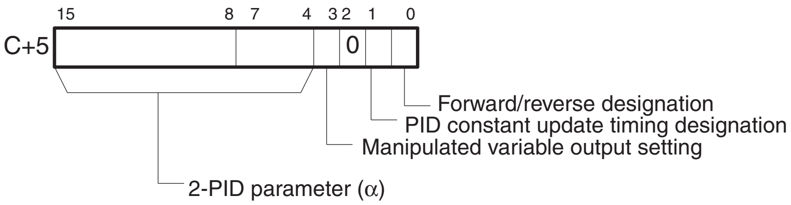
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

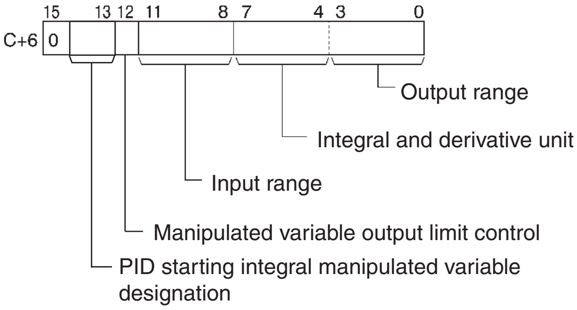
Operands

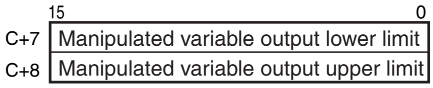
Operand	Description	Data type	Size
S	Input word	UINT	1
C	First parameter word	WORD	41
D	Output word	UINT	1

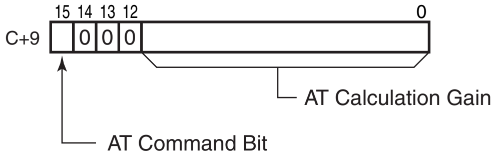
C: First Parameter Word

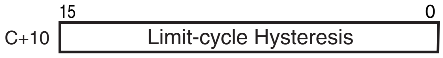


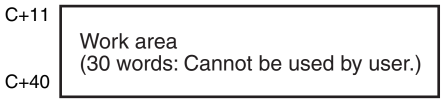












Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
C												---						
D												OK						

Flags

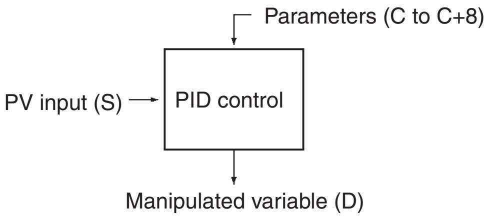
Name	Label	Operation
Error Flag	P_ER	• ON if the C data is out of range. • ON if the actual sampling period is more than twice the designated sampling period. • ON if the manipulated variable output due to the limit cycle does not change for 9,999 seconds during auto tuning. • OFF in all other cases.
Greater Than Flag	P_GT	• ON if the manipulated variable after the PID action exceeds the upper limit. • OFF in all other cases.
Less Than Flag	P_LT	• ON if the manipulated variable after the PID action is below the lower limit. • OFF in all other cases.
Carry Flag	P_CY	• ON while PID control is being executed. • OFF in all other cases.

Function

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:

The AT Command Bit (bit 15 of C+9) is ON at the start of PIDAT(191) execution or it is turned ON during execution.

↓

PID control is interrupted, the PV is forcibly changed, and the PID constants are calculated automatically.

↓

The calculated P, I, and D constants are set in C+1, C+2, and C+3 respectively. The AT Command Bit is turned OFF.

↓

PID control starts (or restarts) with the new PID constants.

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.

- Integral constant (Tik) = &9999 (no integral operation) if the integral constant (Tik) is less than 1 as a result of AT execution with integral and derivative unit = "1: Sampling period multiple". Likewise, differential constant (Tdk) = 0 (no derivative operation) if the derivative constant (Tdk) is less than 1 as a result of AT execution with integral and derivative unit = "1: Sampling period multiple".

Hint

- 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 *3-19-1 PID CONTROL: PID(190)* for details on PID control operations and examples.
- 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.

Precautions

- If PIDAT(191) is used between IL(002) and ILC(003), between JMP(004) and JME(005), or in a subroutine, operation will not be consistent depending on the input condition.
- A PID parameter storage word cannot be shared by multiple PIDAT instructions. Even when the same parameter is used in multiple PIDAT instructions, separate words must be specified.
- 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.)

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 increases.	0001 to 1FFF hex (1 to 8191); (0000 = Derivative operation not executed) (See note 1.)	

Control data	Item	Contents	Setting range	Change with ON input condition
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: $\alpha = 0.65$ Setting from 100 to 163 hex means that the value of the rightmost two digits is set from $\alpha = 0.00$ to $\alpha = 0.99$. (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
Bit 00 of C+5	PID forward/reverse designation	Determines the direction of the proportional action.	0: Reverse action 1: Forward action	Not allowed
Bits 13 to 14 of C+6	ID starting integral manipulated variable designation (See note 4.)	Determines the initial integral manipulated variable when PID control is started (i.e., when the input turns ON) .	Bit 14 = 0 and bit 13=0: Start from same integral manipulated value as manipulated variable output designation (Pre-Ver. 4.0 operation). Bit 14 = 0 or 1 and bit 13 = 1: Bumpless operation (i.e., start from an integral manipulated variable that will not abruptly change the manipulated variable output and result in a continuous change). Bit 14 = 1 and bit 13 = 0: Start with integral manipulated variable = 0.	
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 1: 9 bits 2: 10 bits 3: 11 bits 4: 12 bits 5: 13 bits 6: 14 bits 7: 15 bits 8: 16 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 1: 9 bits 2: 10 bits 3: 11 bits 4: 12 bits 5: 13 bits 6: 14 bits 7: 15 bits 8: 16 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. If the PID parameter is changed after AT is started, PID calculation is started using the value set at the time of the AT interruption.	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
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.

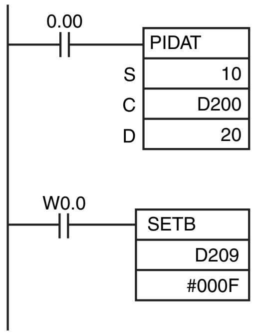
2 Setting the 2-PID parameter (α) to 000 yields 0.65, the normal value.

3 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

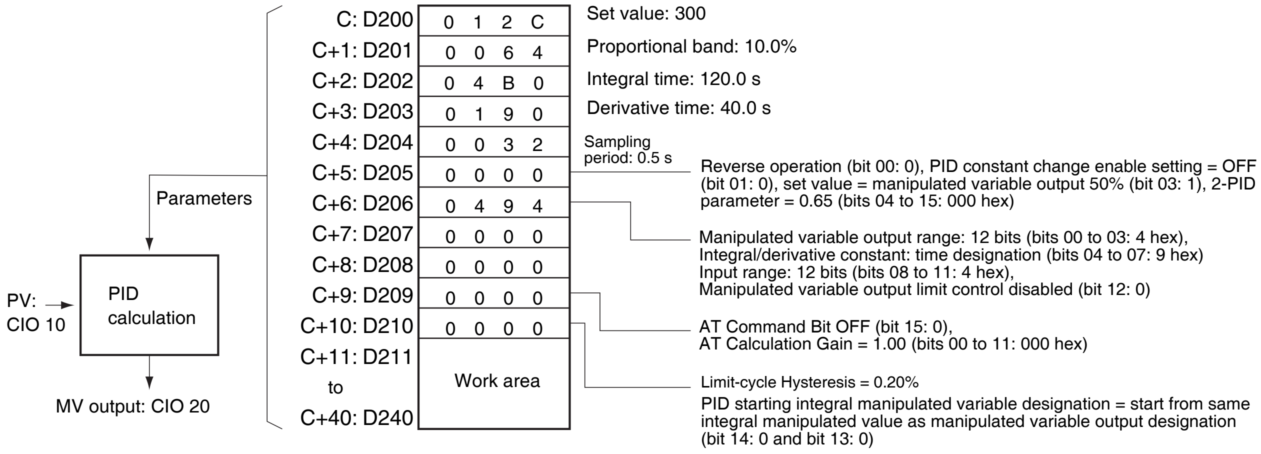
4 CS1-H CPU Units, CJ1-H CPU Units, CJ1M CPU Units, and CS1D CPU Units (for Single CPU Systems) with unit version 4.0 or later, and CJ2 CPU Units only.

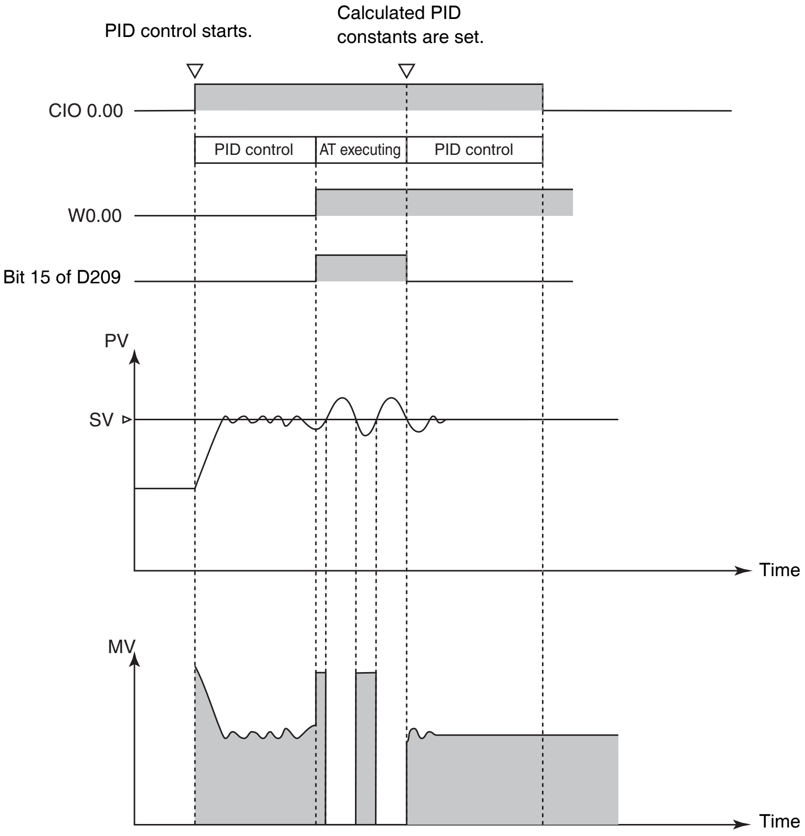
Example Programming

● Interrupting PID Control to Perform Autotuning

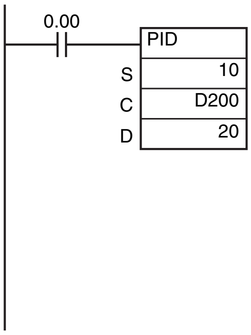


- At the rising edge of CIO 0.00 (OFF to ON), the work area in D211 to D240 is initialized according to the parameters (shown below) set in D200 to D208. After the work area has been initialized, PID control is executed and the manipulated variable is output to CIO 20.
- While CIO 0.00 is ON, PID control is executed at the sampling period intervals according to the parameters set in D200 to D210. The manipulated variable is output to CIO 20.
- 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 0.0 (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.

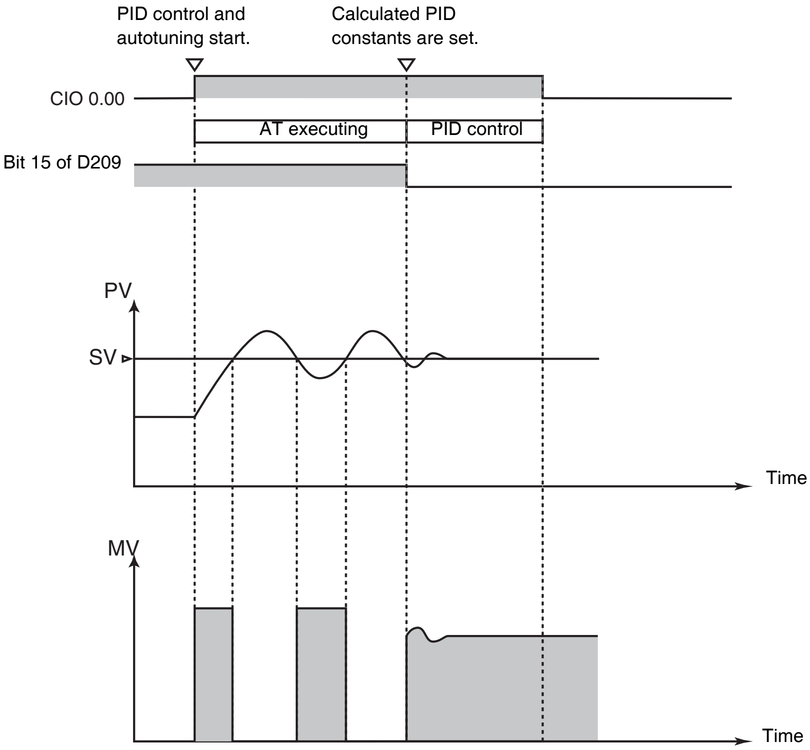



PID control starts.
Calculated PID constants are set.
CIO 0.00
PID control
AT executing
PID control
W0.00
Bit 15 of D209
PV
SV
Time
MV
Time

Starting PIDAT(191) with Autotuning

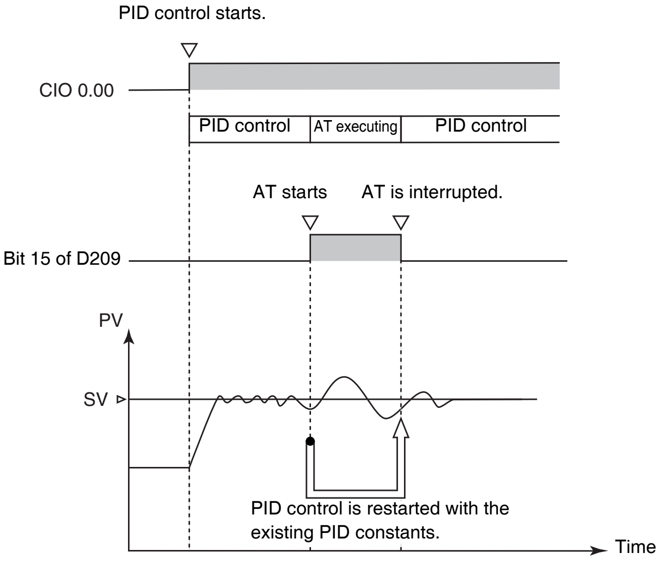


At the rising edge of CIO 0.00 (OFF to ON), autotuning will be performed first if bit 15 of D209 (C+9) is ON. When autotuning is completed, the calculated P, I, and D constants are written to C+1, C+2, and C+3. PID control is then started with the calculated PID constants.



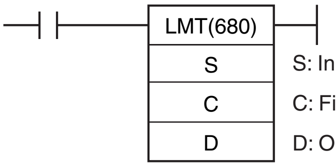
Interrupting Autotuning Before Completion

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.



LMT

Instruction	Mnemonic	Variations	Function code	Function
LIMIT CONTROL	LMT	@LMT	680	Controls output data according to whether or not input data is within upper and lower limits.

	LMT
Symbol	 S: Input word C: First limit word D: Output word

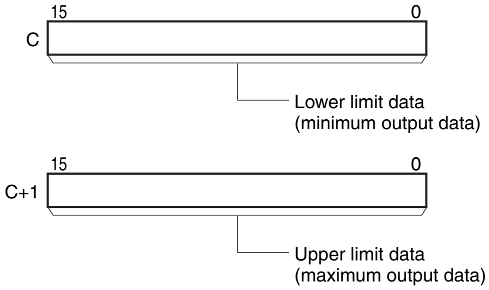
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Input word	INT	1
C	First limit word	DINT	2
D	Output word	INT	1

C: First Limit Word



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C											---	---						
D												OK						

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.

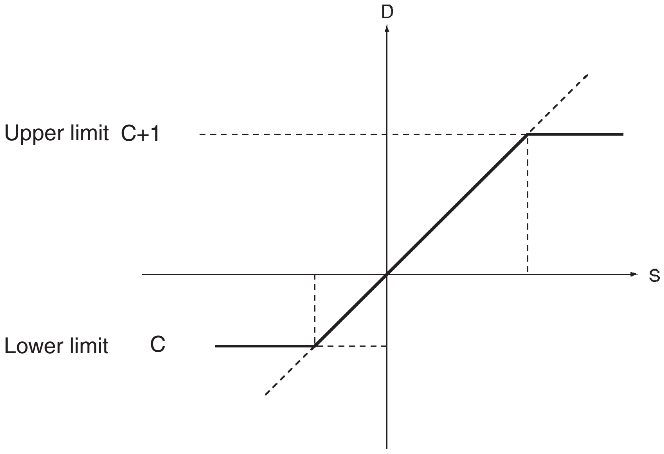
Function

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.

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.

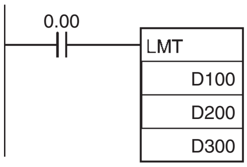


Example Programming

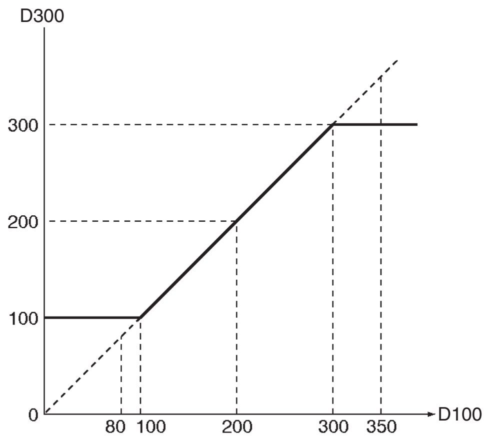
If D100 is 0050 hex (80), then 0064 hex (100) will be output to D300 because 80 is less than the lower limit of 100.

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.

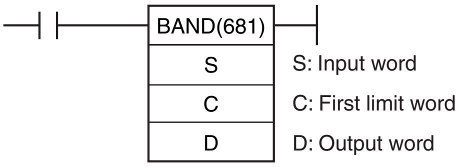


C: D200	0	0	6	4	Lower limit: 100
D201	0	1	2	C	Upper limit: 300



BAND

Instruction	Mnemonic	Variations	Function code	Function
DEAD BAND CONTROL	BAND	@BAND	681	Controls output data according to whether or not input data is within the lower and upper limits of the range (dead band range.)

	BAND
Symbol	

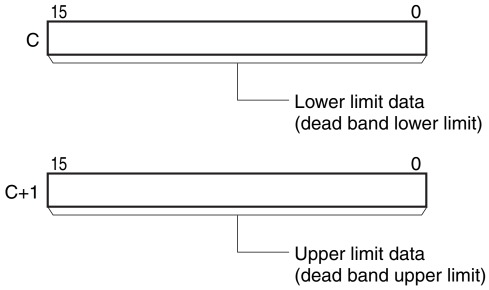
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Input word	INT	1
C	First limit word	UINT	2
D	Output word	UINT	1

C: First Limit Word



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C											---	---						
D												OK						

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.

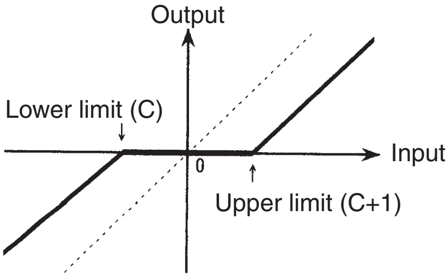
Function

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).

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.



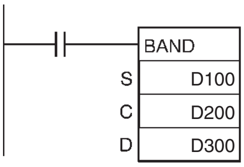
If the output data is smaller than the 8000 (hex) or if is greater than 7FFF, the sign will be reversed. For example, for a lower limit of 0100 (hex) and input data of 8000 (hex), the output data will be as follows: 8000 (hex) [–32768] – 0100 (hex) [256] = 7F00 (hex) [32512]

Example Programming

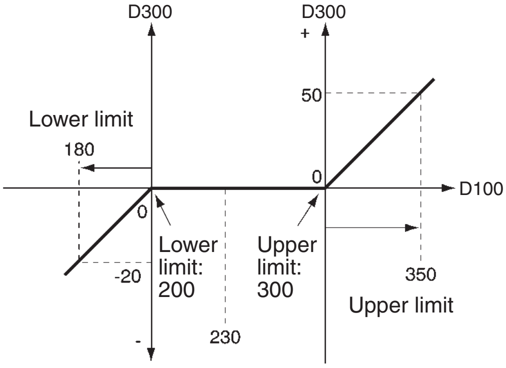
If D100 is 00B4 hex (180), then 180–200=FFEC hex (–20) will be output to D300 because 180 is less than the lower limit of 200.

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.

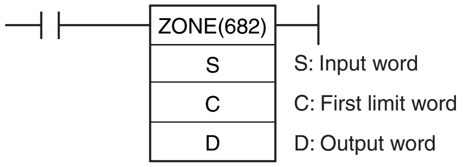


C: D200	0	0	C	8	Lower limit: 200
D201	0	1	2	C	Upper limit: 300



ZONE

Instruction	Mnemonic	Variations	Function code	Function
DEAD ZONE CONTROL	ZONE	@ZONE	682	Adds the specified bias to input data and outputs the result.

	ZONE
Symbol	

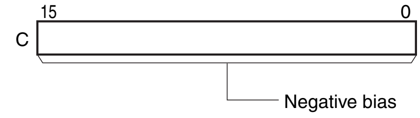
Applicable Program Areas

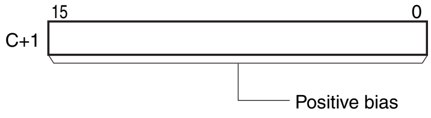
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Input word	INT	1
C	First limit word	UDINT	2
D	Output word	UINT	1

C: First Limit Word





● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C											---	---						
D												OK						

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.

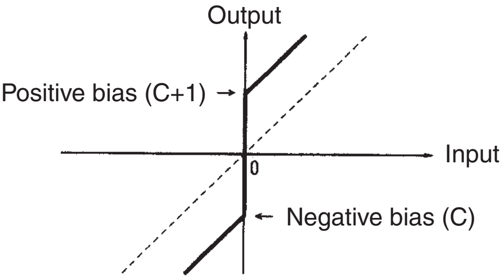
Function

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.

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, 0 will be output to D and the Equals Flag will turn ON.



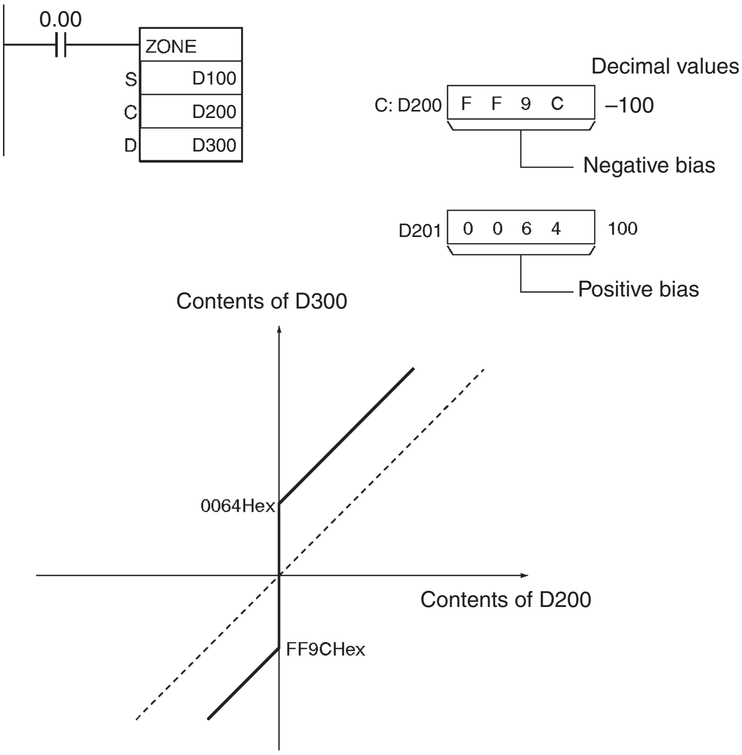
If the output data is smaller than the 8000 (hex) or if is greater than 7FFF, the sign will be reversed. For example, for a negative bias value of FF00 (hex) and input data of 8000 (hex), the output data will be as follows:
8000 (hex) [–32768] – FF00 (hex) [–256] = 7F00 (hex) [32512]

Example Programming

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.

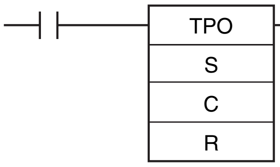
If the value of D100 is 0, then 0000 hex will be stored in D300.

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.



TPO

Instruction	Mnemonic	Variations	Function code	Function
TIME-PROPORTIONAL OUTPUT	TPO	---	685	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.

	TPO
Symbol	 S: Input word C: First parameter word R: Pulse output bit

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Input word	UINT	1
C	First parameter word	WORD	7
R	Pulse output bit	BOOL	---

S: Input Word

Specifies the input word containing the input duty ratio or manipulated variable.

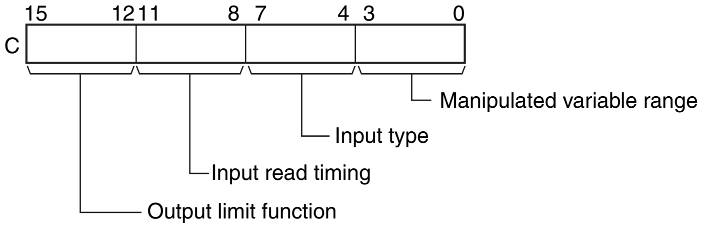
- Input duty ratio: 0000 to 2710 hex (0.00% to 100.00%)
- Input manipulated variable (See note.): 0000 to FFFF hex (0 to 65,535 max.) (Bits 00 to 03 of C specify the manipulated variable range, i.e., the number of valid bits in the manipulated variable. Specify the same number of bits as specified for the output range setting in PID(190).)

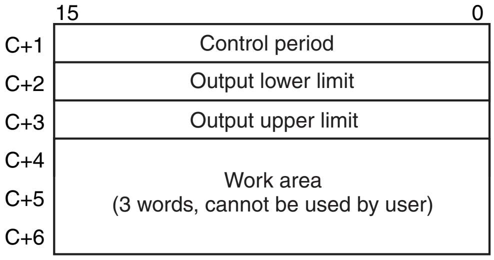
Note If S is a manipulated variable, specify the word containing the manipulated variable output from a PID(190) or PIDAT(191) instruction.

C: First Parameter Word

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.)

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C											---	---						
R					---	---	---	---	---	---								

Flags

Name	Label	Operation
Error Flag	P_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.

Function

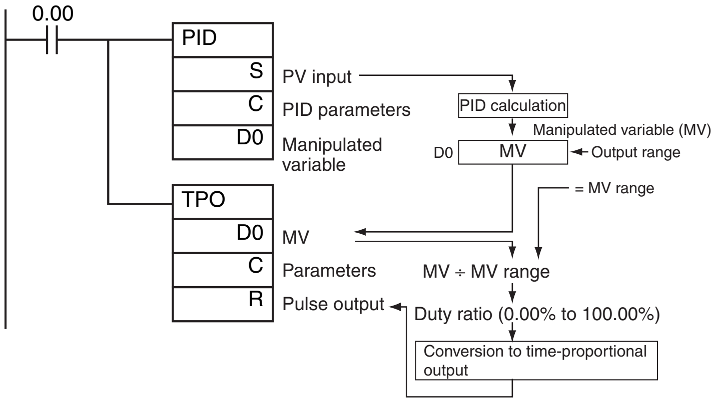
Receives a duty ratio or manipulated variable input from the word address specified by S, converts the duty ratio to a time-proportional output (see note) based on the parameters specified in words C to C+3, and outputs a pulse output to the bit specified by R.

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

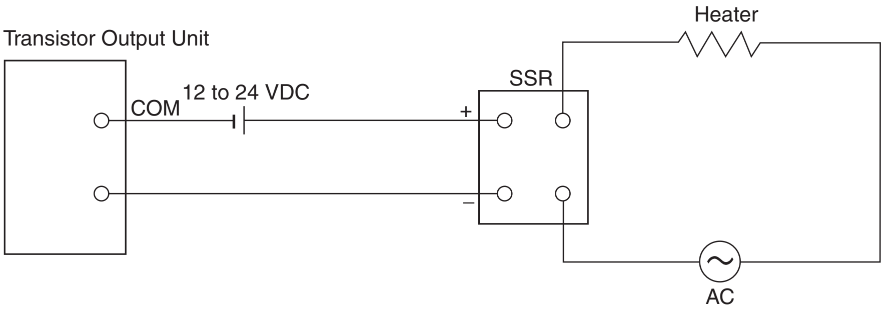
When combining TPO(685) with a PID control instruction, the manipulated variable input is divided by the manipulated variable range to calculate the duty ratio, that duty ratio is converted to a time-proportional output, and pulses are output.



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 1 hex: 9 bits 2 hex: 10 bits 3 hex: 11 bits 4 hex: 12 bits 5 hex: 13 bits 6 hex: 14 bits 7 hex: 15 bits 8 hex: 16 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

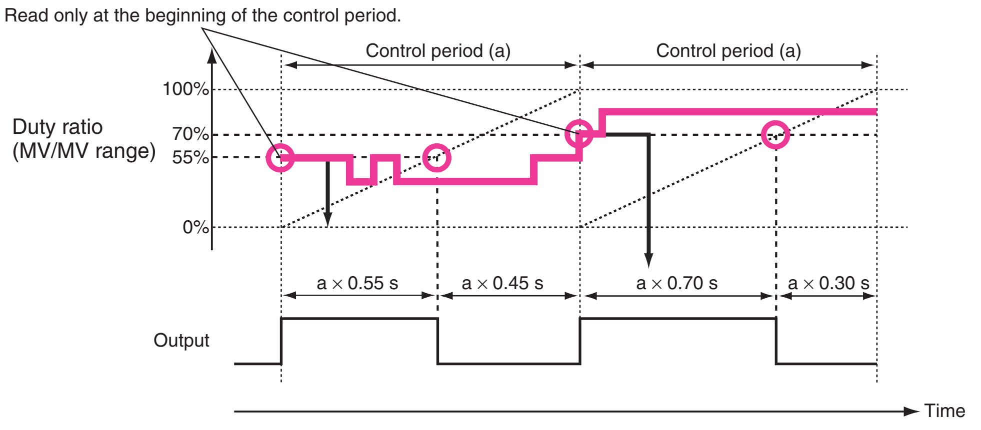
- The instruction is executed while the input condition is ON.
- When instruction execution starts, the output bit (R) is turned ON/OFF according to the duty ratio.
- The parameters (in C to C+3) are read in real time each time that the instruction is executed. When changing the parameters, change all 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).
- 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.

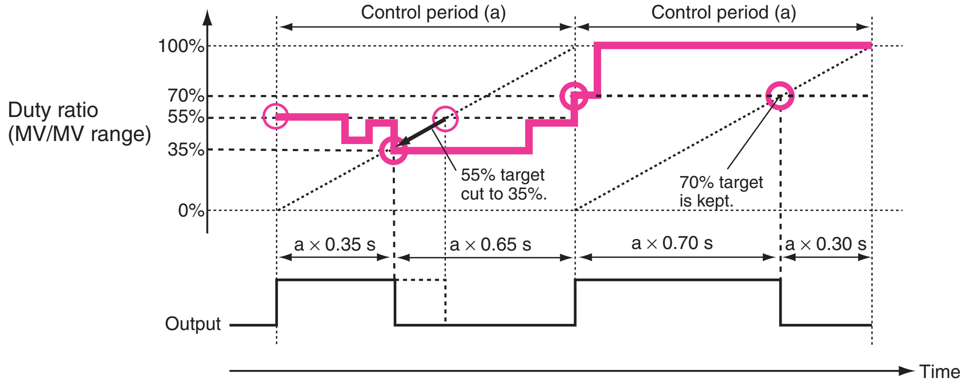
The following diagrams show the operation of each input read timing setting.

- Input time setting = 0 (Use the beginning value of the control period.)



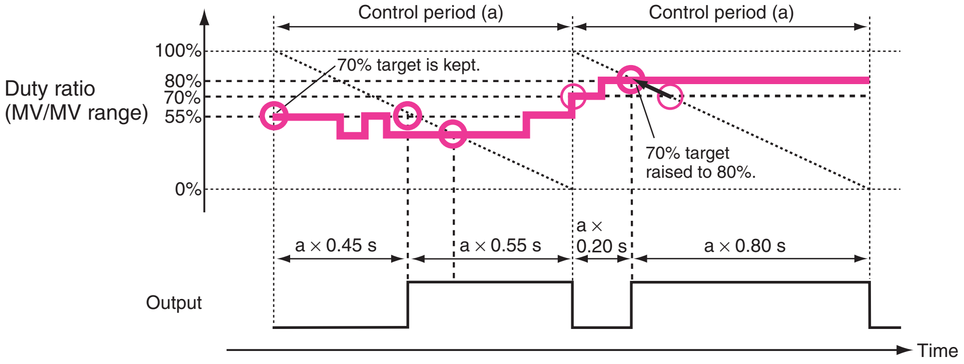
Each control period's output is determined by the duty ratio at the beginning of that period.
Use this setting for general applications.

- Input time setting = 1 (Use lower value.)



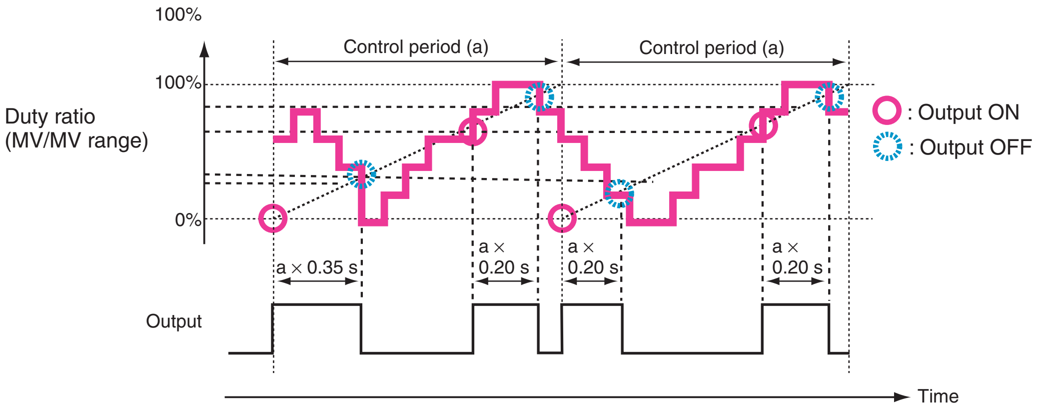
If the duty ratio falls below the initial value early enough, the duty ratio will be adjusted and the output will be turned OFF sooner.
Use this setting for applications such as avoiding overshooting when using time-proportional control to control heating and using a relatively long control period.

- 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)

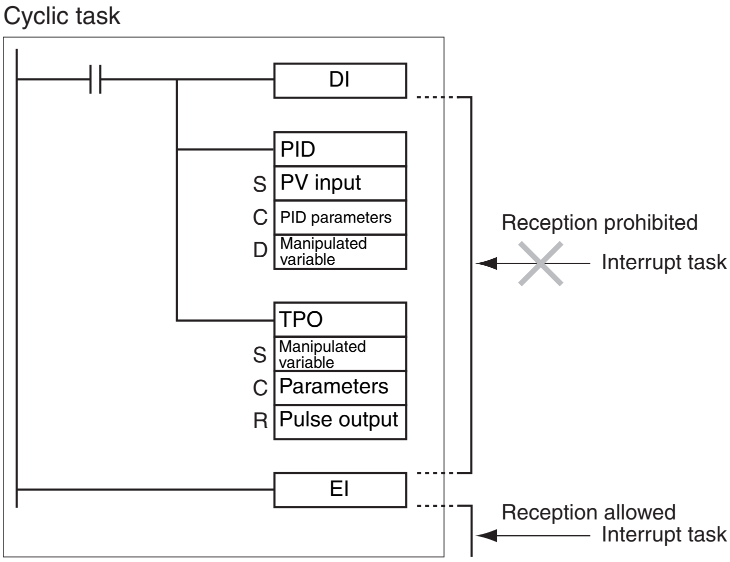


Changes in the duty ratio are monitored in real time. If the duty ratio falls below the initial value early enough, the duty ratio will be adjusted and the output will be turned OFF sooner. If the duty ratio rises again after that, the ratio will be adjusted again and the output will be turned ON. This 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.

- The output limiter function (bits 12 to 15 of C) can be enabled to restrict (saturate) output when it is outside the range between the output limiter lower limit (C + 2) and output limiter upper limit (C + 3).

Precautions

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.

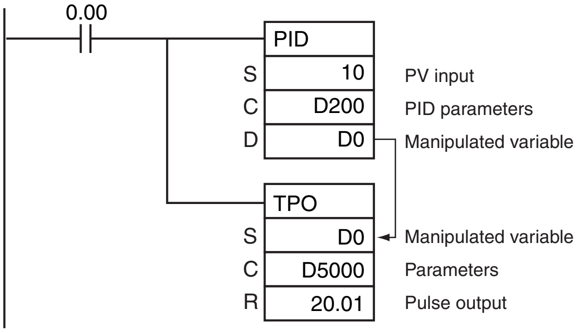


Example Programming

● Combining TPO(685) with PID(190)

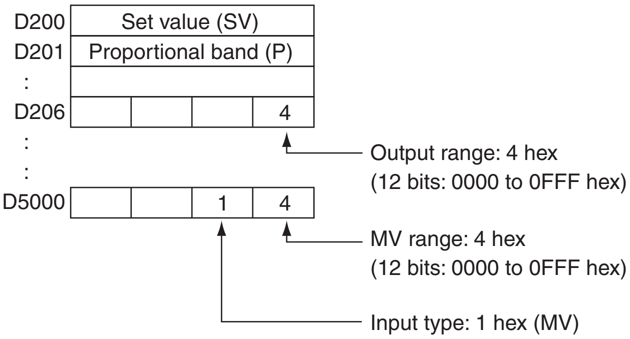
When CIO 0.00 is ON, TPO(685) takes the manipulated variable output from PID(190) (contained in D0), calculates the duty ratio from that manipulated variable value (Duty ratio = MV ÷ MV range), converts the duty ratio to a time-proportional output, and outputs the pulses to CIO 2001.

In this case, CIO 20 is allocated to a Transistor Output Unit and bit CIO 2001 is connected to a solid state relay for heater control.



When CIO 0.00 goes from OFF to ON, PID(190) reads the parameters, performs the PID calculation with the PV input in CIO 10, and outputs the manipulated variable (MV) to D0.

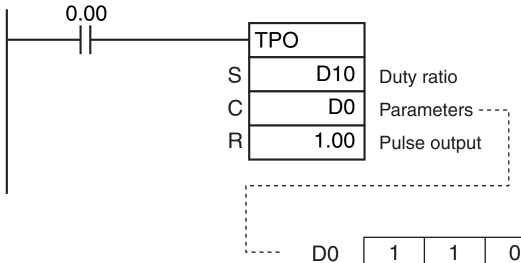
TPO(685) calculates the duty ratio by dividing the MV in D0 by the MV range (0FFF Hex since the range is set to 12 bits), converts that duty ratio to a time-proportional output, and outputs the pulse output to bit 01 of CIO 20.



● 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.

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%.

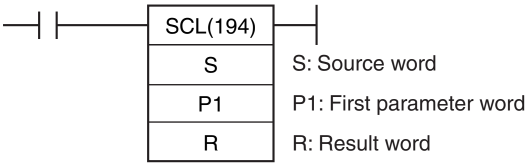


TPO(685) takes the duty ratio in D10, converts that duty ratio to a time-proportional output, and outputs the pulse output to bit 00 of CIO 1.

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%

SCL

Instruction	Mnemonic	Variations	Function code	Function
SCALING	SCL	@SCL	194	Converts unsigned binary data into unsigned BCD data according to the specified linear function.

	SCL
Symbol	 S: Source word P1: First parameter word R: Result word

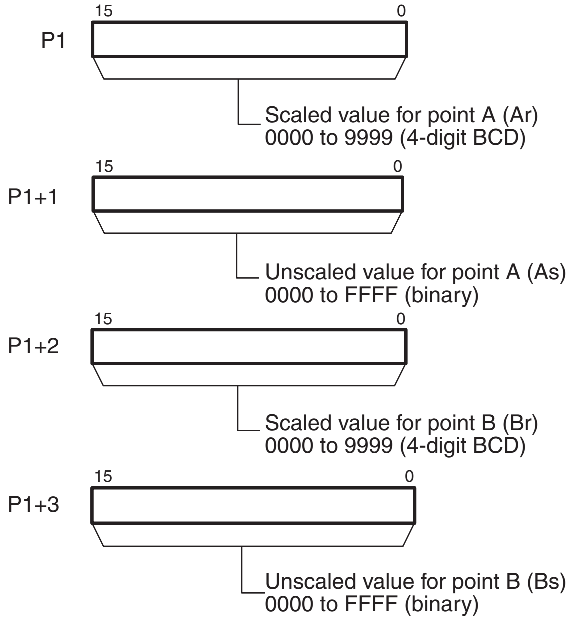
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	UINT	1
P1	First parameter word	LWORD	4
R	Result word	WORD	1

P1: First Parameter Word



Note P1 to P1+3 must be in the same area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
P1												---						
R												OK						

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.

Function

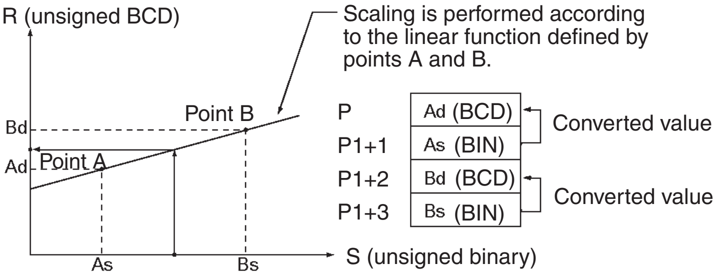
SCL(194) is used to convert the unsigned binary data contained in the source word S into unsigned BCD data and place the result in the result word R according to the linear function defined by points (As, Ad) and (Bs, Bd). The address of the first word containing the coordinates of points (As, Ar) and (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.

$$R = Bd - \frac{(Bd - Ad)}{\text{BCD conversion of } (Bs - As)} \times \text{BCD conversion of } (Bs - S)$$

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.



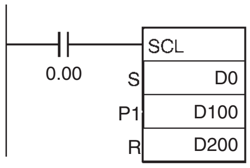
Hint

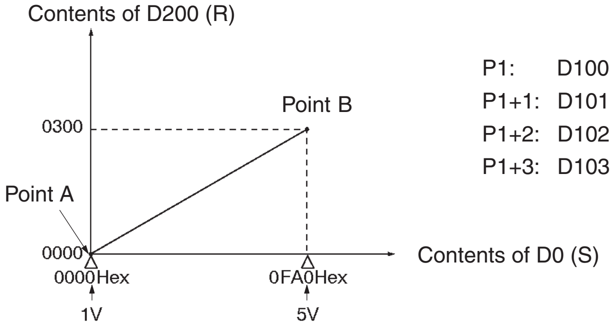
- SCL(194) can be used to scale the results of analog signal conversion values from Analog Input Units according to user-defined scale parameters. For example, if a 1 to 5-V input to an Analog Input Unit is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to 50 to 200°C 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.

Example Programming

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 0 and 300 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.

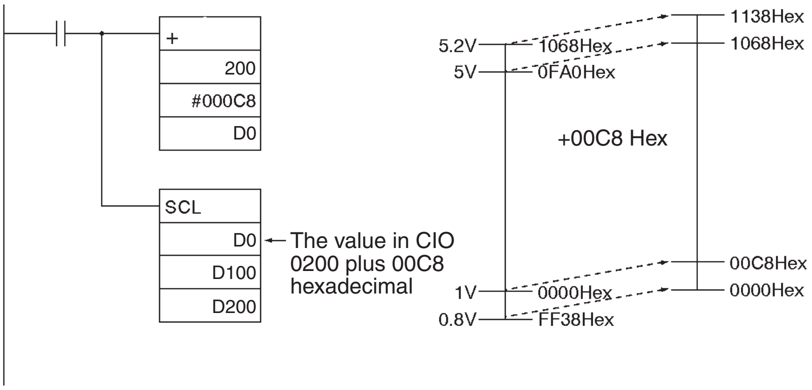


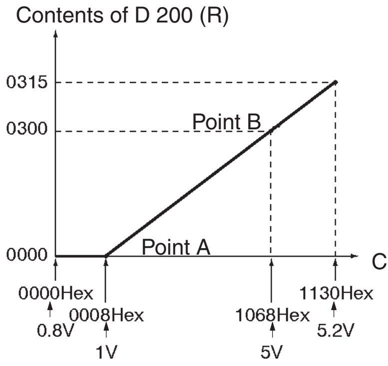


P1:	D100	0 0 0 0	Ad (BCD)
P1+1:	D101	0 0 0 0	As (BIN)
P1+2:	D102	0 3 0 0	Bd (BCD)
P1+3:	D103	0 F A 0	Bs (BIN)

Reference:

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.





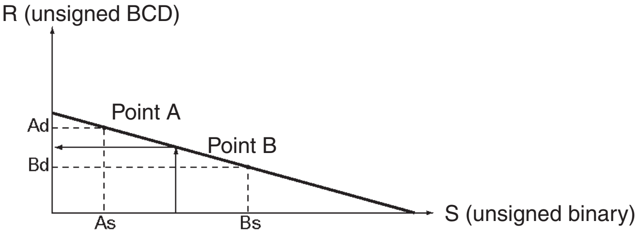
Point A (00C8 Hex → 0000 (BCD))
Point B (1068 Hex → 0300 (BCD))

P1:	D100	0 0 0 0	Ad (BCD)
P1+1:	D101	0 0 C 8	As (BIN)
P1+2:	D102	0 3 0 0	Bd (BCD)
P1+3:	D103	1 0 6 8	Bs (BIN)

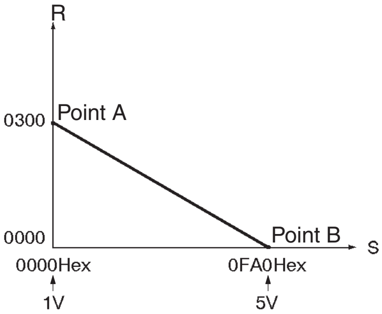
In this example, values from 0000 to 00C8 hexadecimal will be converted to negative values. SCL(194), however, can output only unsigned BCD values from 0000 to 9999, so 0000 BCD will be output whenever the contents of D0.00 is between 0000 and 00C8 hexadecimal.

Reverse Scaling

Reverse scaling can also be used by setting As < Bs and Ar > Br. The following relationship will result.

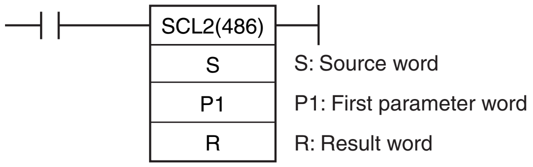


Reverse scaling can be used, for example, to convert (reverse scale) 1 to 5 V (0000 to 0FA0 hexadecimal) to 0300 to 0000, respectively, as shown in the following diagram.



SCL2

Instruction	Mnemonic	Variations	Function code	Function
SCALING 2	SCL2	@SCL2	486	Converts signed binary data into signed BCD data according to the specified linear function. An offset can be input in defining the linear function.

	SCL2
Symbol	

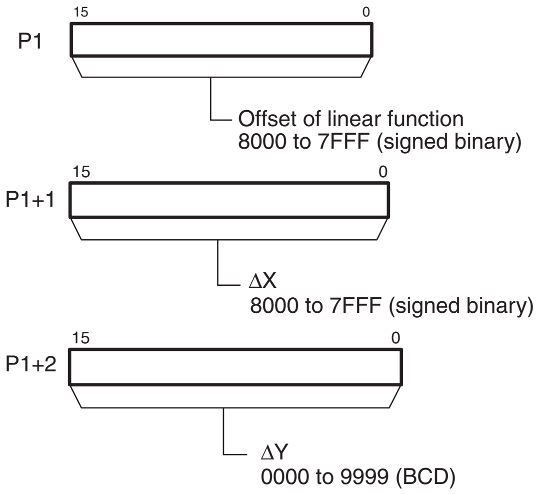
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	INT	1
P1	First parameter word	WORD	3
R	Result word	WORD	1

P1: First Parameter Word



Note P1 to P1+2 must be in the same area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
P1												---						
R												OK						

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.

Function

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.

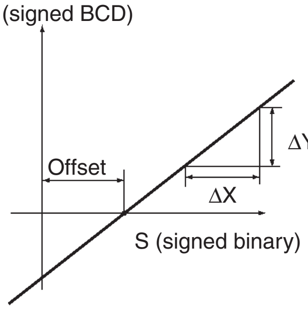
$$R = \frac{\Delta Y}{\text{BCD conversion of } \Delta X} \times ((\text{BCD conversion of S}) - (\text{BCD conversion of offset}))$$

Note: The slope of the line is ΔY/ΔX.

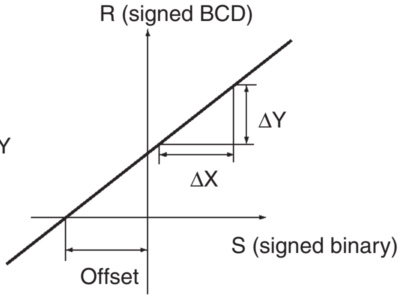
The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

- 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.

Positive Offset

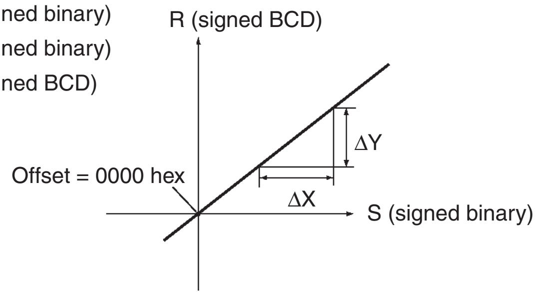


Negative Offset



P1	Offset	(Signed binary)
P1+1	ΔY	(Signed binary)
P1+2	ΔX	(Signed BCD)

Offset of 0000



Hint

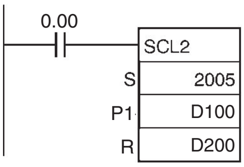
- SCL2(486) can be used to scale the results of analog signal conversion values from Analog Input Units according to user-defined scale parameters. For example, if a 1 to 5-V input to an Analog Input Unit is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to –100 to 200°C 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.

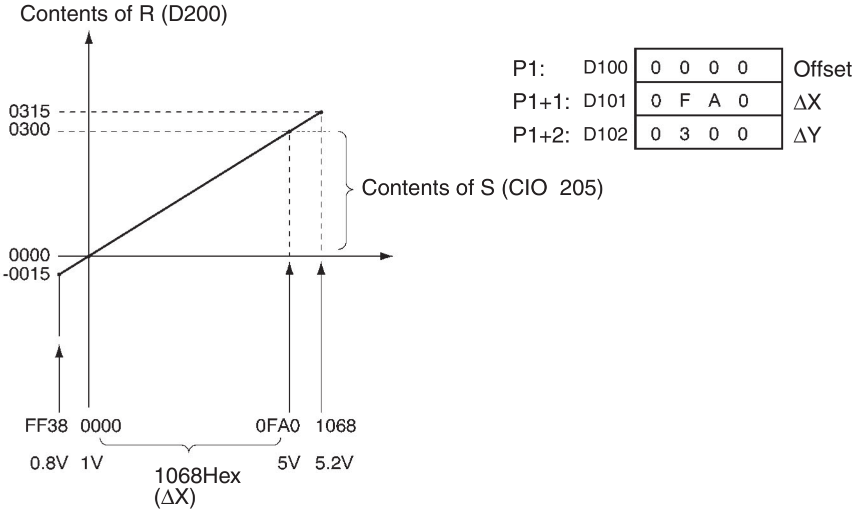
Example Programming

● 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 205 as 0000 to 0FA0 hexadecimal. SCL2(486) is used to convert (scale) the value in CIO 205 to a value between 0000 and 0300 BCD.

When CIO 0.00 is ON, the contents of CIO 205 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.

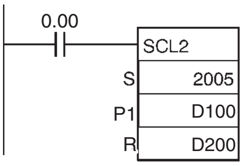


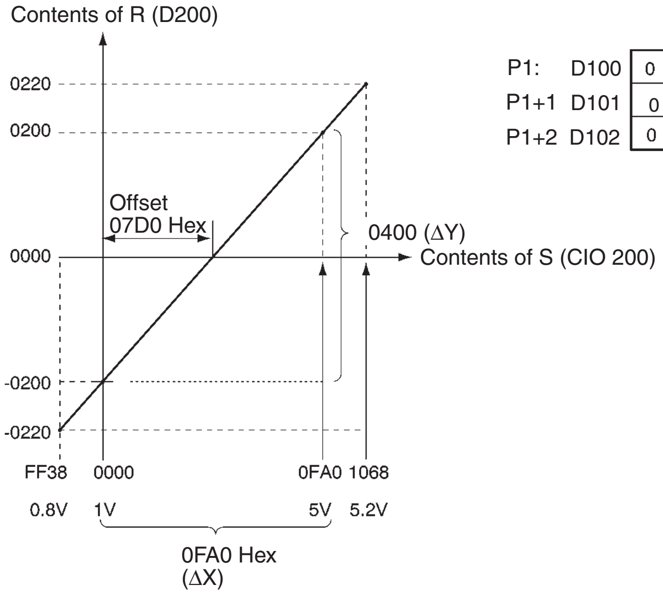


● Scaling 1 to 5-V Analog Input to –200 to 200

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 ∆X (0FA0), ∆Y (0400), and the offset (07D0). These values are contained in D100 to D102, and the result is output to D200.

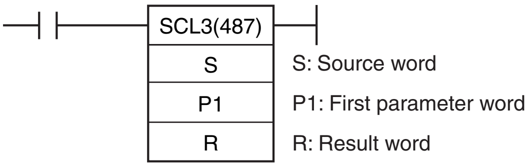




P1:	D100	0 7 D 0	Offset
P1+1	D101	0 F A 0	∆X
P1+2	D102	0 4 0 0	∆Y

SCL3

Instruction	Mnemonic	Variations	Function code	Function
SCALING 3	SCL3	@SCL3	487	Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function.

	SCL3
Symbol	

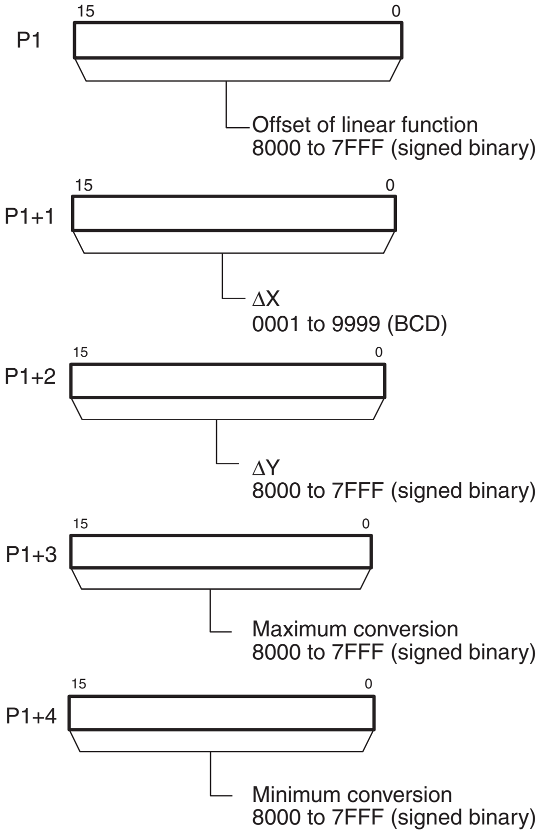
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
P1	First parameter word	WORD	5
R	Result word	INT	1

P1: First Parameter Word



Note P1 to P1+4 must be in the same area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
P1												---						
R												OK						

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.

Function

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.

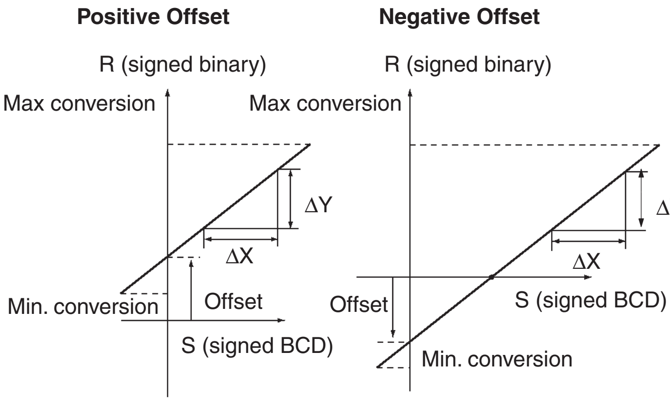
The sign of the result is indicated by the status of the Carry Flag (ON: negative, OFF: positive). Use STC(040) and CLC(041) to turn the Carry Flag ON and OFF.

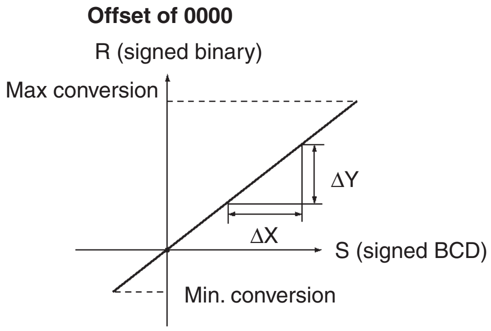
The following equations are used for the conversion.

$$R = \frac{\Delta Y}{\text{Binary conversion of } \Delta X} \times ((\text{Binary conversion of S})+(\text{Offset}))$$

Note: 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.



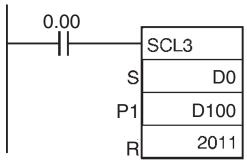


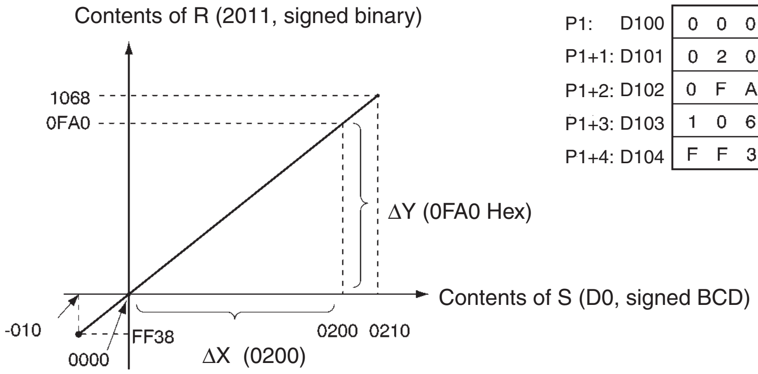
Hint

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.

Example Programming

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.

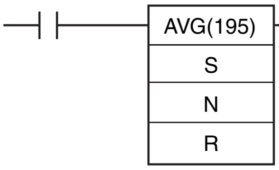




P1: D100	0 0 0 0	Offset
P1+1: D101	0 2 0 0	ΔX
P1+2: D102	0 F A 0	ΔY
P1+3: D103	1 0 6 8	Max. conversion
P1+4: D104	F F 3 8	Min. conversion

AVG

Instruction	Mnemonic	Variations	Function code	Function
AVERAGE	AVG	---	195	Calculates the average value of an input word for the specified number of cycles.

	AVG
Symbol	 S: Source word N: Number of cycles R: Result first word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

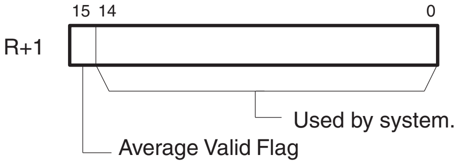
Operand	Description	Data type	Size
S	Source word	UINT	1
N	Number of cycles	UINT	1
R	Result first word	UINT	Variable

N: Number of Cycles

The number of cycles must be between 0001 and 0040 hexadecimal (0 to 64 cycles).

R: Result First Word and R+1: First Work Area Word

R: Average
R+1: Processing information



Average Valid Flag
OFF: Not valid (AVG(195) has not yet been executed the specified number of cycles.)
ON: Valid.

R+2: Previous value #1
⋮
R+N+1: Previous value #N

Note R to R+N+1 must be in the same area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S, N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
R											---	---						

Flags

Name	Label	Operation
Error Flag	ER	• ON if the contents of N is 0. • OFF in all other cases.

Function

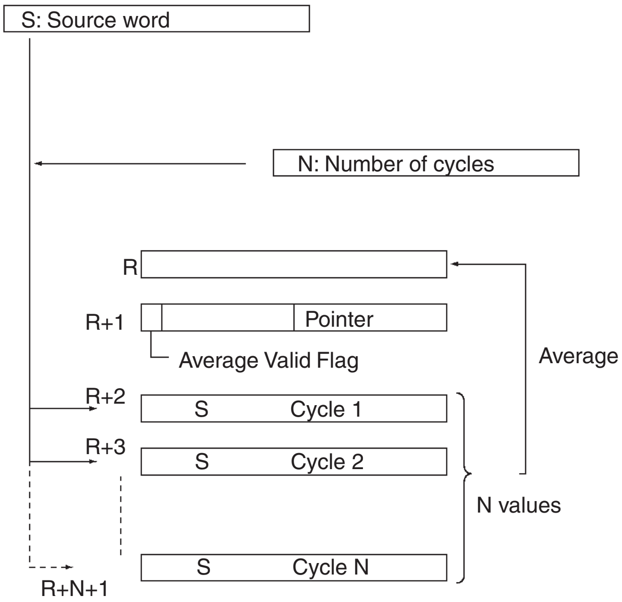
For the first N–1 cycles when the execution condition is ON, AVG(195) writes the values of S in order to words starting with R+2. The Previous Value Pointer (bits 00 to 07 of R+1) is incremented each time a value is written. Until the Nth value is written, the contents of S will be output unchanged to 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. If a value greater than 64 is specified, operation will use a value of 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.



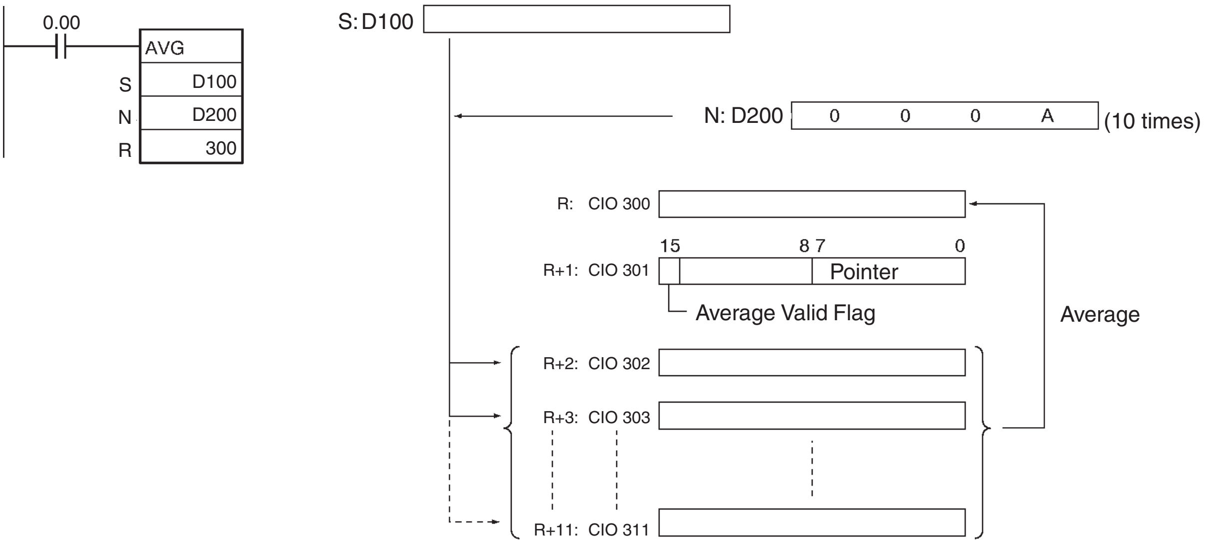
Precautions

The processing information (R+1) is cleared to 0000 each time the execution condition changes from OFF to ON.

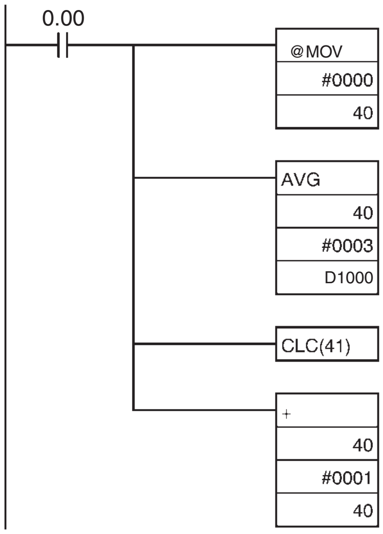
But the processing information (R+1) will not be cleared to 0000 the first time the program is executed at the start of operation. If AVG(195) is to be executed in the first program scan, clear the First Work Area Word from the program.

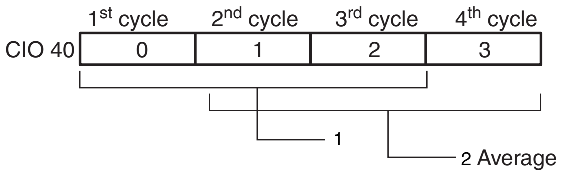
Example Programming

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.



- In the following example, the content of CIO 40 is set to #0000 and then incremented by 1 each cycle.
- For the first two cycles, AVG(195) moves the content of CIO 40 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.





D1000	0	1	1	2	Average
D1001	1	2	8000	8001	Pointer
D1002	0	0	0	3	3 previous values of IR 40
D1003	---	1	1	1	
D1004	---	---	2	2	

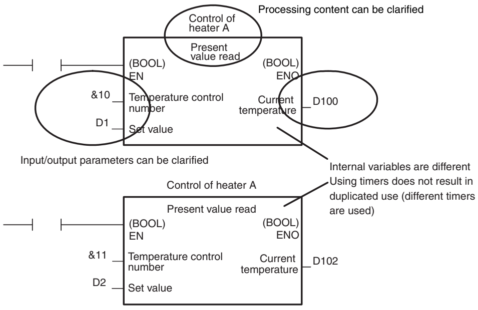
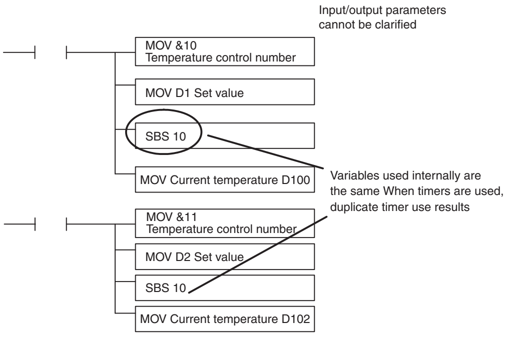
Subroutines

Subroutine instruction

In the CS/CJ Series, function blocks and subroutines can be used as a means of structuring programs.

The advantages and disadvantages of each are as follows:

● Differences between function blocks and subroutines

	Function blocks	Subroutines
Unit versions of CPU Units that can be used	CS1-H, CJ1-H, CJ1M CPU Units: Unit version 3.0 or later CS1D CPU Units: Unit version 4.0 or later CJ2 CPU Units: All unit versions	All unit versions
Process names	Instance names can be assigned.	Names cannot be assigned
Clarification of input and output	Clarification as FB input variable or output variable is possible	Clarification is not possible
Variable names	Names can be assigned to variables	Variable names can be assigned
Internal variables	Yes When instance names are the same, the same variable is accessed. When instance names are different, different variables are accessed. As such, data can be stored separately for each call (instance) in a function block. Using timers in function blocks does not result in duplicated use.	No All subroutines are accessed using the same variables. For this reason, data cannot be stored separately for each call in a subroutine. If timers are used in a subroutine, duplicated timer use will result.
Program writing method	Variable programming by ladder or ST language	Address programming by ladder
Indication by diagram	Processing content and input/output parameters are easier to understand (visibility can be heightened)  	Address programming by ladder 
Protection	Read protect can be set for each function block (Supported from CX-Programmer 6.1)	No protect function
Storing	Can be stored by function block	Storing is not possible by subroutine.

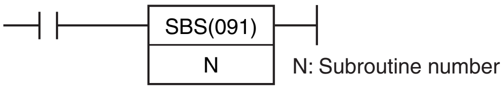
The subroutine function has two types of subroutines: regular subroutines and global subroutines. The differences between the two types are as follows.

● Differences between regular subroutines and global subroutines

	Regular subroutines	Global subroutines
Functions	A subroutine within the same task can be called. For example, a subroutine in task 0 cannot be called from task 1.	Global subroutines can be called from all tasks. A subroutine in interrupt task 0 can be called from any task.
Combined-use instructions	SBS (subroutine call) instruction SBN (subroutine entry) instruction RET (subroutine return) instruction	GSBS (global subroutine call) instruction GSBN (global subroutine entry) instruction GRET (global subroutine return) instruction
Entry in subroutine program	Entered at the end of the task that uses the subroutine (before the END instruction).	Entered at the end of interrupt task 0 (before the END instruction).

SBS

Instruction	Mnemonic	Variations	Function code	Function
SUBROUTINE CALL	SBS	@SBS	091	Calls the subroutine with the specified subroutine number and executes that program.

	SBS
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
N	Subroutine number	---	1

N: Subroutine number

Specifies the subroutine number between 0 and 1023 decimal.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the subroutine number must be between the range &0 to &255 decimal.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Combined-use instructions

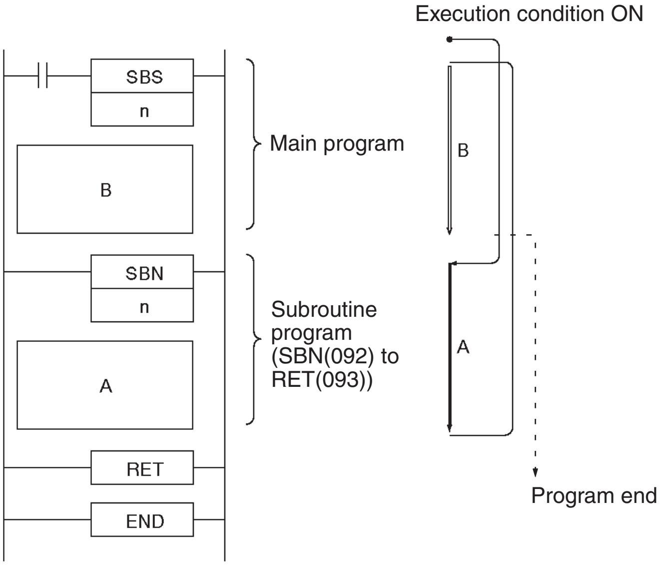
SBN (subroutine entry) instructions and RET (subroutine return) instructions

Flags

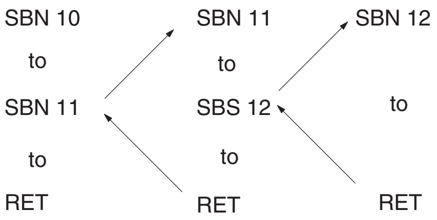
Name	Label	Operation
Error Flag	ER	• ON if nesting exceeds 16 levels. • ON if the specified subroutine number does not exist. • ON if a subroutine calls itself. • ON if a subroutine being executed is called. • ON if the specified subroutine is not defined in the current task. • OFF in all other cases.

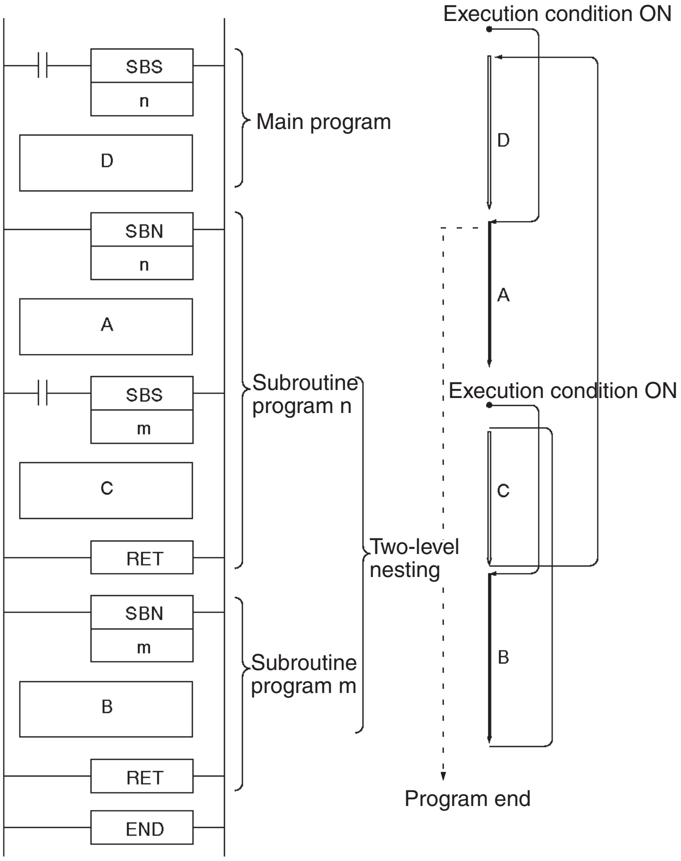
Function

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).
A subroutine can be called more than once in a program.



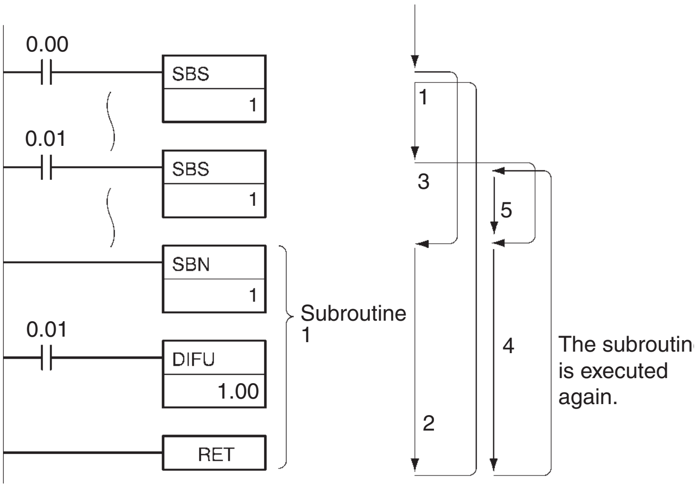
Subroutines can be nested up to 16 levels. Nesting is when another subroutine is called from within a subroutine program, such as shown in the following example, which is nested to 3 levels.

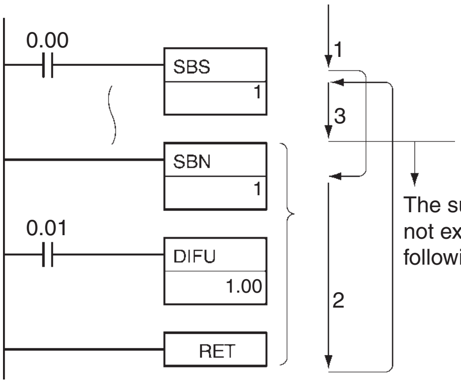




Precautions

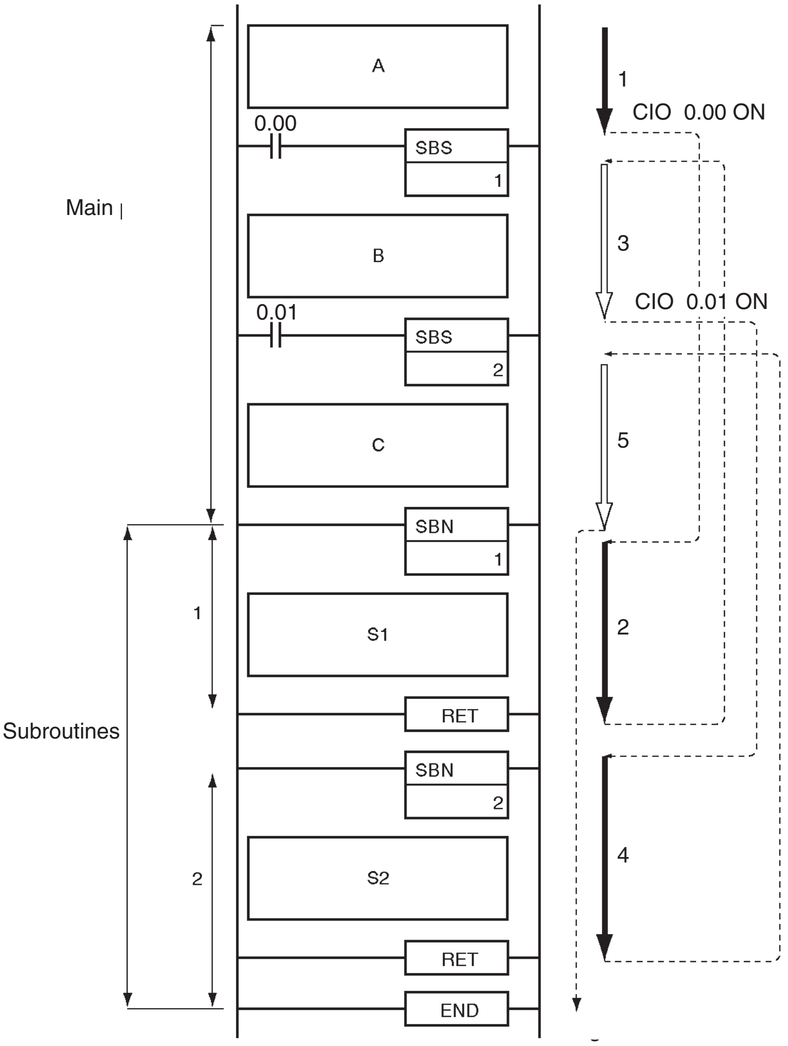
- The subroutine number must be unique for each subroutine. You cannot use the same number for more than one subroutine.
- Each subroutine must have a unique subroutine number. Do not use the same subroutine number for more than one subroutine.
- Observe the following precautions when using differentiated instructions (DIFU(013), DIFD(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 1 is executed when CIO 0.00 is ON and CIO 1.00 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 1 will be executed again but this time DIFU(013) will turn CIO 1.00 OFF without checking the status of CIO 0.01.
  - In contrast, a differentiated instruction (UP, DOWN, DIFU(013) or DIFD(014)) would maintain the ON status 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 1 is executed if CIO 0.00 is ON. Output CIO 1.00 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 1 will not be executed again and output CIO 1.00 will remain ON
  - SBS(091) will be treated as NOP(000) when it is within a program section interlocked by IL(002) and ILC(003).





Example Programming

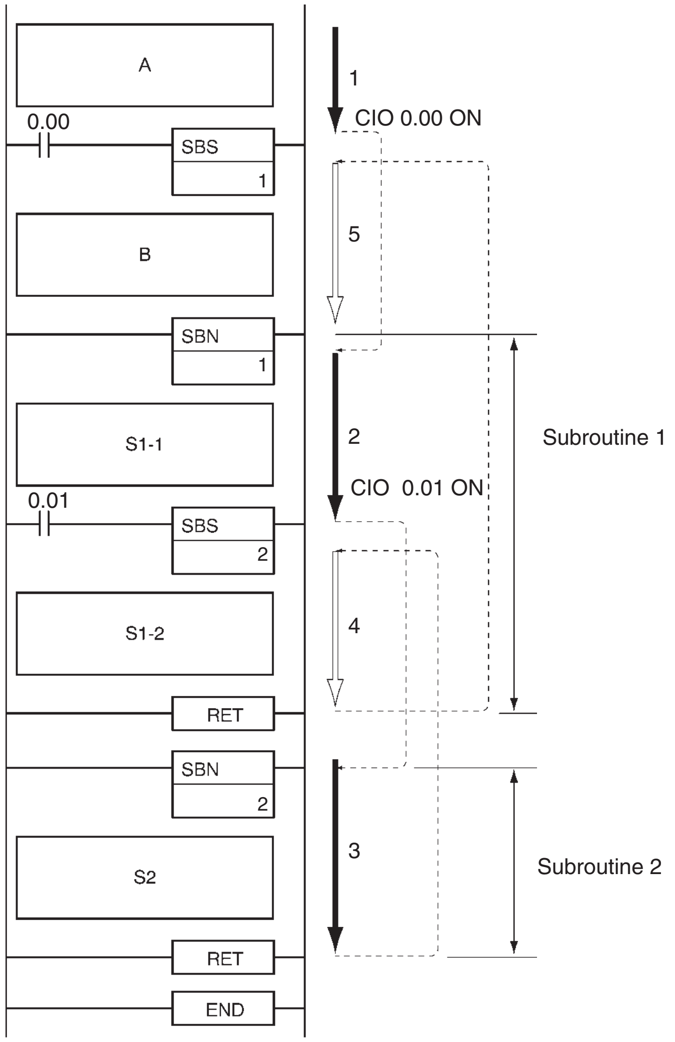
● Sequential (Non-nested) Subroutines



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

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.

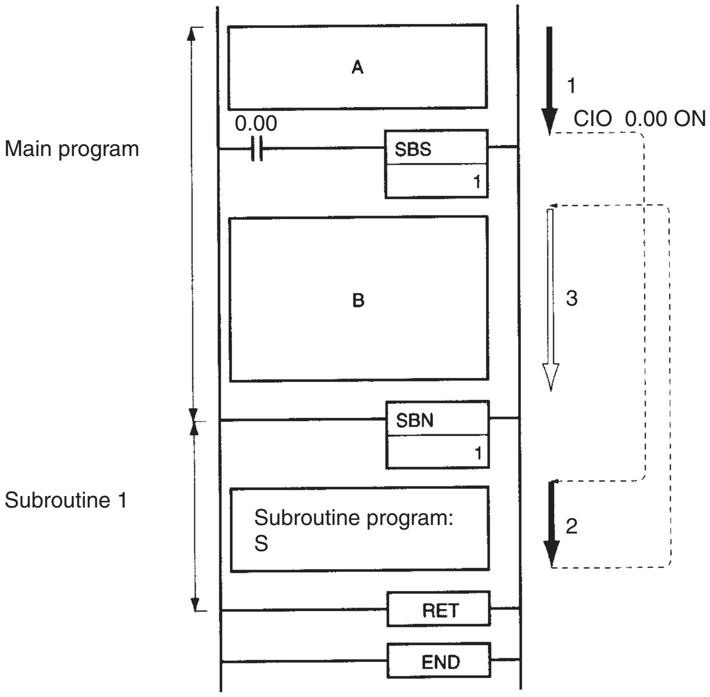
Nested Subroutines



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

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.

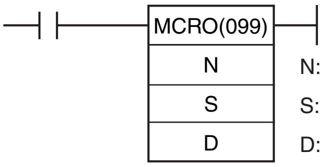
When CIO 0.00 is ON in the following example, subroutine 1 is executed and program execution returns to the next instruction after SBS(091). The remainder of the main program (through the instruction just before SBN(092) 1) is then executed.



0.00	Order of execution
ON	A→S→B
OFF	A→B

MCRO

Instruction	Mnemonic	Variations	Function code	Function
MACRO	MCRO	@MCRO	099	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.

	MCRO
Symbol	 N: Subroutine number S: First input parameter word D: First output parameter word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
N	Subroutine number	---	1
S	First input parameter word	---	4
D	First output parameter word	---	4

N: Subroutine number

Specifies the subroutine number between 0 and 1023 decimal.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the subroutine number must be between the range 0 to 255 decimal.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---			OK				
D																		

Combined-use instructions

SBN (subroutine entry) instructions and RET (subroutine return) instructions

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.

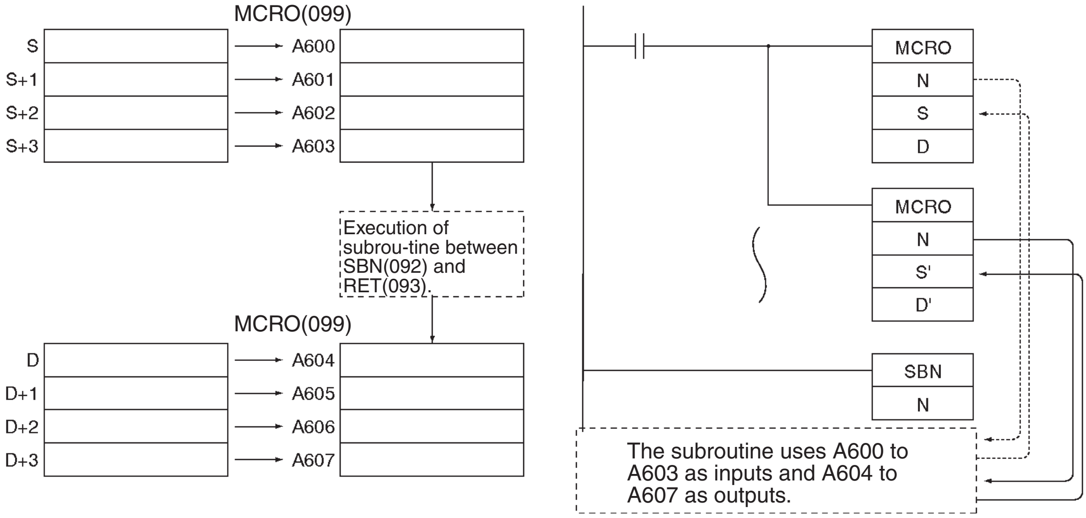
Related Auxiliary Area Words and Bits

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 output words	A604 to A607	After the subroutine specified in MCRO(099) has been executed, the output data in these output words are copied to D to D+3.

Function

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).



Hint

When MCRO(099) is executed, the specified input and output data is transferred to the specified subroutine.
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.
However, only subroutines that differ by address but are otherwise exactly the same can be consolidated by a MCRO (macro) instruction.

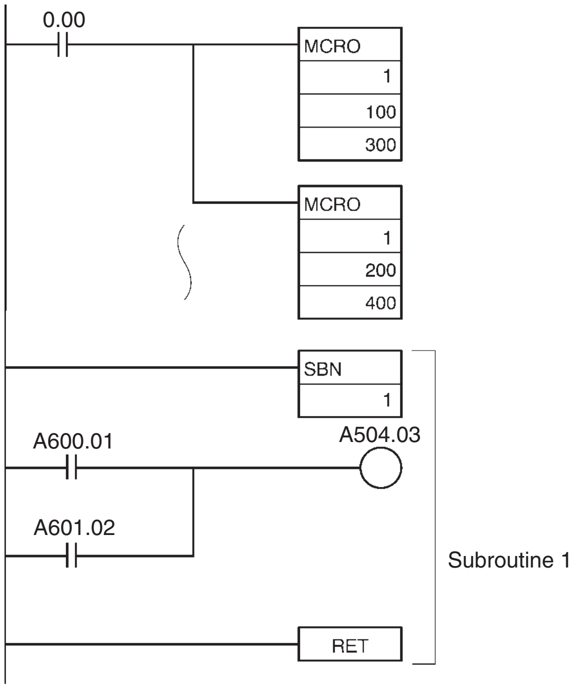
Precautions

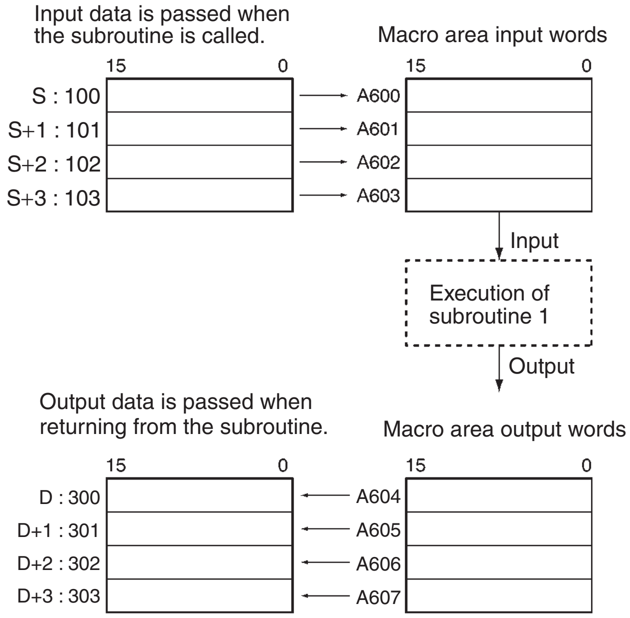
- The four words of input data (words or bits) in A600 to A603 and the four words of output data (words or bits) in A604 to A607 must be used in the subroutine called by MCRO(099). It is not possible to pass more than four words of data.
- It is possible to nest MCRO(099) instructions, but the data in 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.
- When a MCRO instruction is used in an interrupt task, there is a possibility that the input words (A600 to A603C) and output words (A604 to A607CH) will be overwritten. For this reason, programming to save the data is necessary.

Example Programming

When CIO 0.00 is ON in the following example, two MCRO(099) instructions pass different input and output data to subroutine 1.

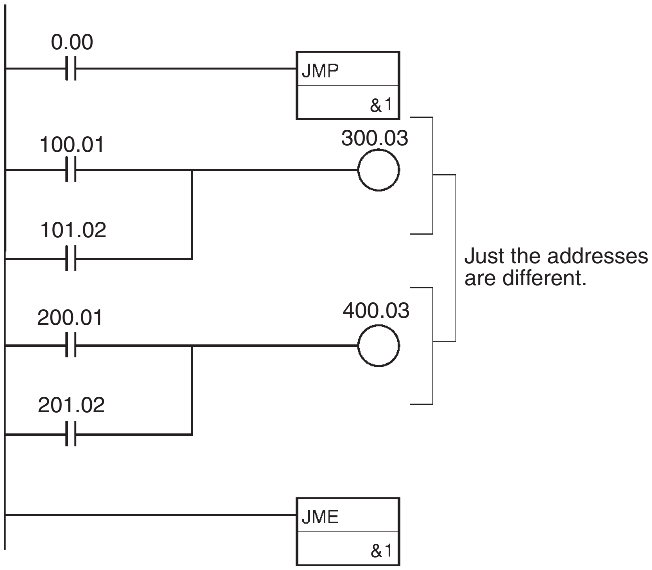
1. The first MCRO(099) instruction passes the input data in CIO 100 to CIO 103 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 300 to CIO 303.
2. The second MCRO(099) instruction passes the input data in CIO 200 to CIO 203 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 400 to CIO 403.





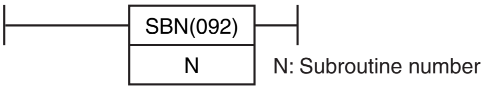
The second MCRO(099) instruction operates in the same way, but the input data in CIO 200 to CIO 203 is passed to A600 to A603 and the output data in A604 to A607 is passed to CIO 400 to CIO 403.

|| (Equivalent)



SBN/RET

Instruction	Mnemonic	Variations	Function code	Function
SUBROUTINE ENTRY	SBN	---	092	Indicates the beginning of the subroutine program with the specified subroutine number.
SUBROUTINE RETURN	RET	---	093	Indicates the end of a subroutine program.

	SBN	RET
Symbol	 SBN(092) N N: Subroutine number	 RET(093)

Applicable Program Areas

● SBN

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	Not allowed	Not allowed	OK	Not allowed

● RET

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	Not allowed	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
		SBN	
N	Subroutine number	---	1

● SBN
N: Subroutine number

Specifies the subroutine number between 0 and 1023 decimal.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the subroutine number must be between the range 0 to 255 decimal.

● Operand Specifications

Area		Word addresses								Indirect DM/EM addresses		Con- stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
SBN	N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Combined-use instructions

SBS (subroutine call) instruction or MCRO (macro) instruction

Flags

● SBN/RET

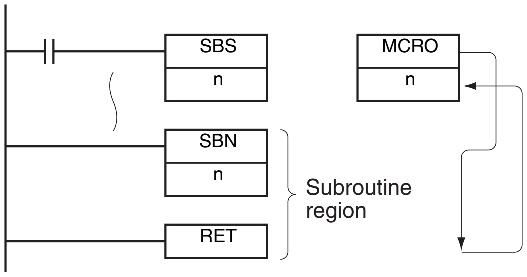
There are no flags affected by this instruction.

Function

● SBN

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).

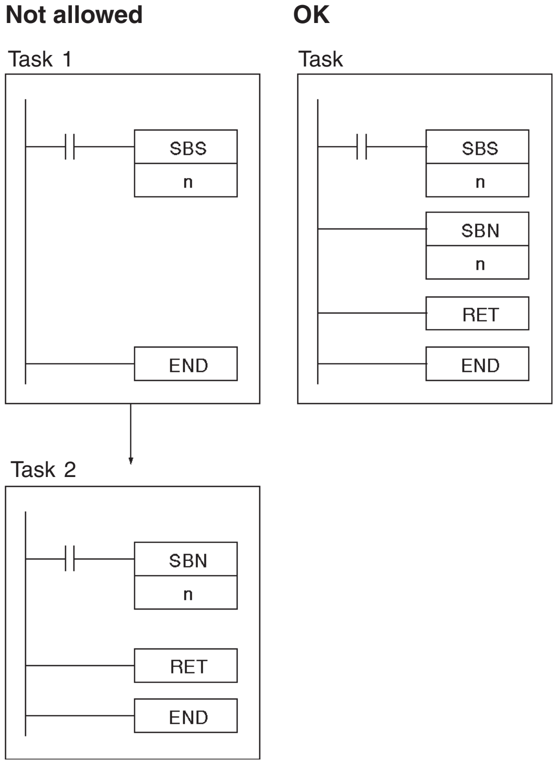


● RET

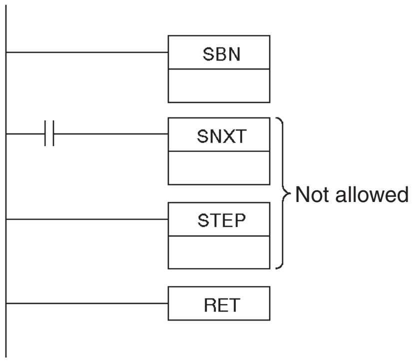
When program execution reaches RET(093), it is automatically returned to the next instruction after the SBS(091) or MCRO(099) instruction that called the subroutine. When the subroutine has been called by MCRO(099), the output data in A604 through A607 is written to D through D+3 before program execution is returned.

Precautions

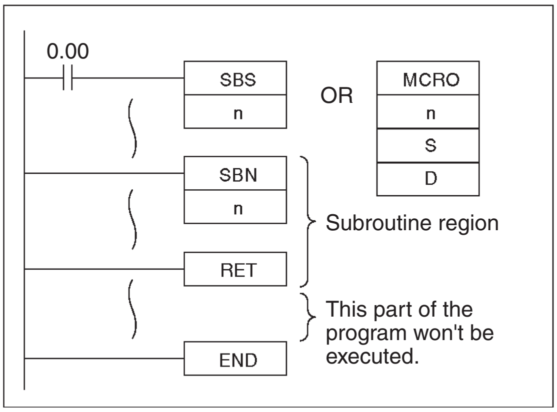
- Place the subroutine program area (SBN(092) to RET(093)) in the same task as the SBS(091) or MCRO(099) instruction of the same number. Subroutines in other tasks cannot be called.



- The step instructions, STEP(008) and SNXT(009) cannot be used in subroutines.



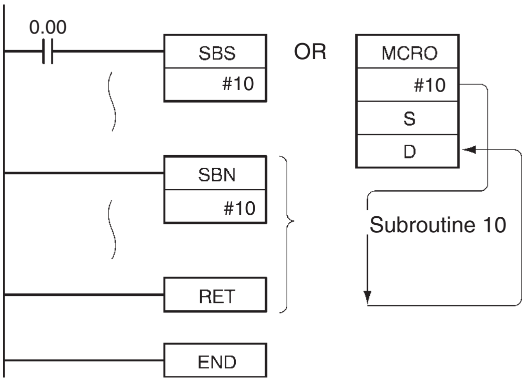
- Place the subroutines after the main program and just before the END(001) instruction in the program for each task. If part of the main program is placed after the subroutine region, that program section will be ignored.



Note : The input method for the subroutine number, N, is different for the CX-Programmer and a Programming Console. Input #0 to #1023 on the CX-Programmer and 0000 to 1023 on a Programming Console.

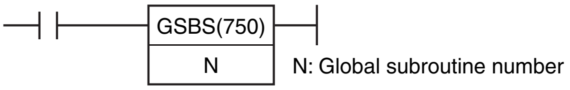
Example Programming

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.



GSBS

Instruction	Mnemonic	Variations	Function code	Function
GLOBAL SUBROUTINE CALL	GSBS	@GSBS	750	Calls the global subroutine with the specified subroutine number and executes that program.

	GSBS
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
		SBN	
N	Global subroutine number	---	1

N: Subroutine number

Specifies the subroutine number between 0 and 1023 decimal.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the subroutine number must be between the range 0 to 255 decimal.

Note Global subroutine numbers and regular subroutine numbers (SBS instructions and MCRO instruction operands) are shared (do not use the same number for a global subroutine and a regular subroutine).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Combined-use instructions

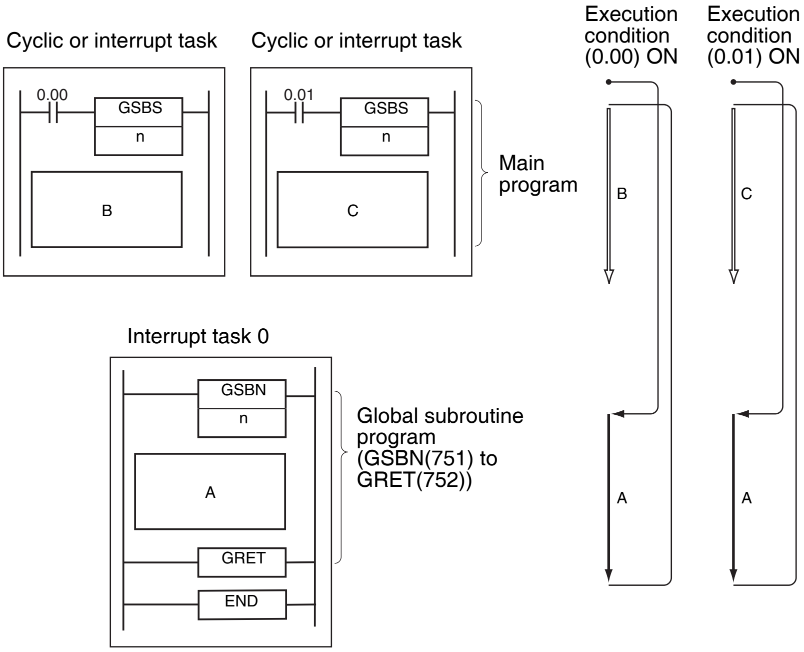
GSBS(750) is used in combination with GSBN(751) and GRET(752), the GLOBAL SUBROUTINE ENTRY and GLOBAL SUBROUTINE RETURN instructions.

Flags

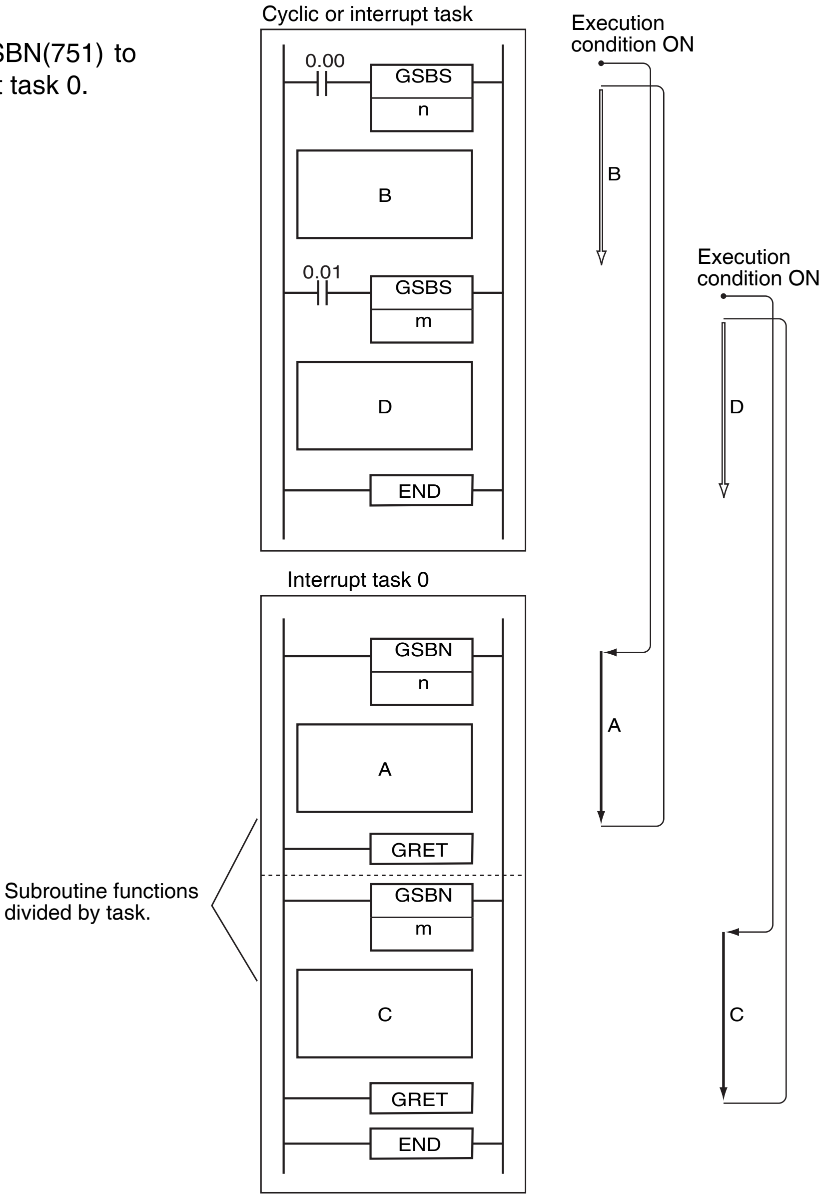
Name	Label	Operation
Error Flag	P_ER	• ON if nesting exceeds 16 levels (counting both regular and global subroutines). • ON if the specified global subroutine does not exist. • ON if a global subroutine calls itself. • ON if a global subroutine being executed is called. • ON if the specified subroutine is not defined in interrupt task 0. • OFF in all other cases.

Function

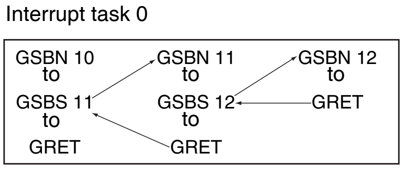
GSBS(750) calls the global subroutine with the specified global subroutine number. The global subroutine is the program section between GSBN(751) and GRET(752). When the global subroutine is completed, program execution continues with the next instruction after GSBS(750).
The same global subroutine region (GSBN(751) to GRET(752)) can be called more than once.
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.



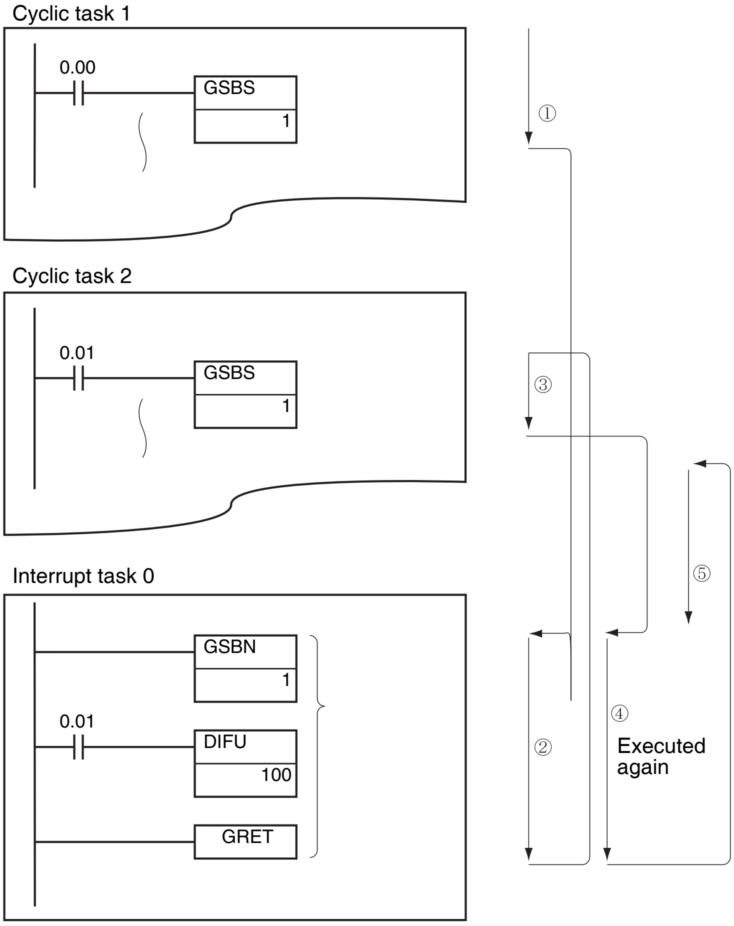
Precautions

The regular SUBROUTINE CALL instruction, SBS(091), cannot call a global subroutine region (GSBN(751) to GRET(752)).

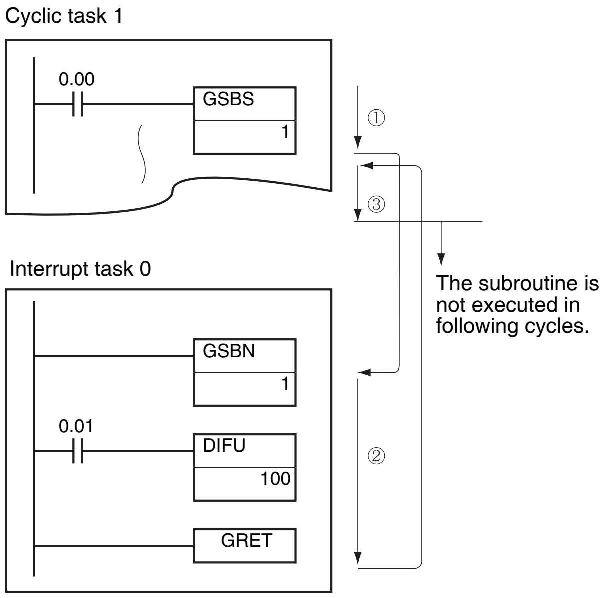
GSBS(750) will not be executed when it is within a program section interlocked by IL(002) and ILC(003), so interlocks are not allowed within global subroutine regions.

- Observe the following precautions when using differentiated instructions (UP, DOWN, DIFU(013), DIFD(014), or up/down differentiated instructions) in subroutines.

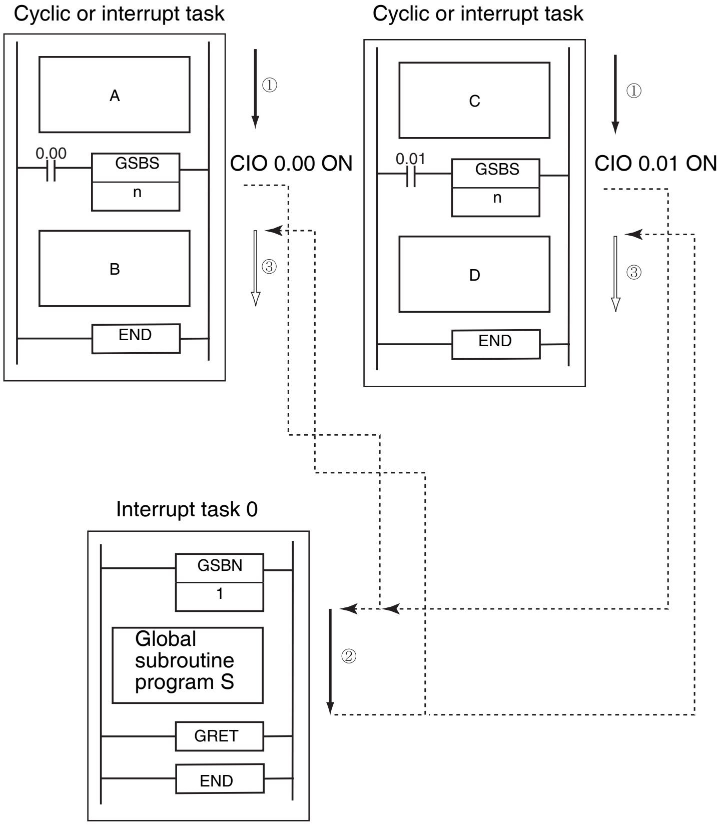
  The operation of differentiated instructions in a global subroutine is unpredictable if a subroutine is executed more than once in the same cycle. In the following example, global subroutine 1 is executed when CIO 0.00 is ON and CIO 1.00 is turned ON by DIFU(013) when CIO 0.01 has gone from OFF to ON. If CIO 0.01 is ON in the same cycle, global subroutine 1 will be executed again but this time DIFU(013) will not detect the rising edge of CIO 0.01 and CIO 1.00 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 1 is executed if CIO 0.00 is ON. Output CIO 1.00 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 1 will not be executed again and output CIO 1.00 will remain ON.



Example Programming



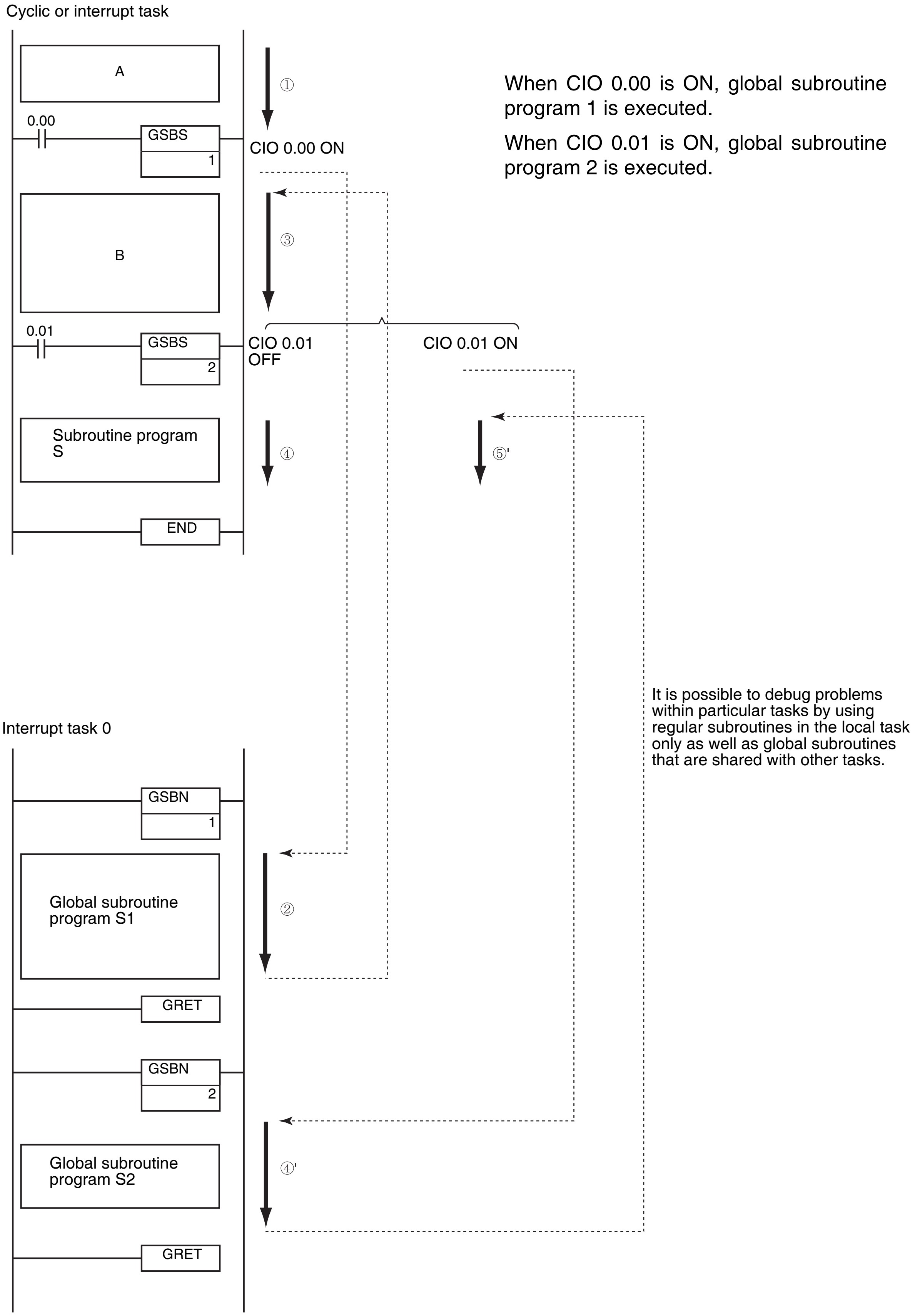
When CIO 0.00 is ON in the following example, global subroutine 1 is executed and program execution returns to the next instruction after GSBS(750).

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

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.



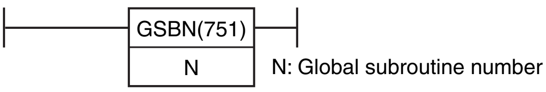
When CIO 0.00 is ON, global subroutine program 1 is executed.

When CIO 0.01 is ON, global subroutine program 2 is executed.

It is possible to debug problems within particular tasks by using regular subroutines in the local task only as well as global subroutines that are shared with other tasks.

GSBN/GRET

Instruction	Mnemonic	Variations	Function code	Function
GLOBAL SUBROUTINE ENTRY	GSBN	---	751	Indicates the beginning of the global subroutine program with the specified subroutine number.
GLOBAL SUBROUTINE RETURN	GRET	---	752	Indicates the end of a subroutine program.

	GSBN	GRET
Symbol	 N: Global subroutine number	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	Not allowed	Not allowed	OK	Not allowed

Operands

Operand	Description	Data type	Size
		GSBN	
N	Global subroutine number	---	1

N: Subroutine number

Specifies the subroutine number between 0 and 1023 decimal.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the subroutine number must be between the range 0 to 255 decimal.

Note Global subroutine numbers and regular subroutine numbers (SBS instructions or MCRO instruction operands) are shared (do not use the same number for a global subroutine and a regular subroutine).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Combined-use instructions

● GSBN/GRET

GSBN(751) and GRET(752) are used in combination with GSBS(750), the GLOBAL SUBROUTINE CALL.

Flag

● GSBN/GRET

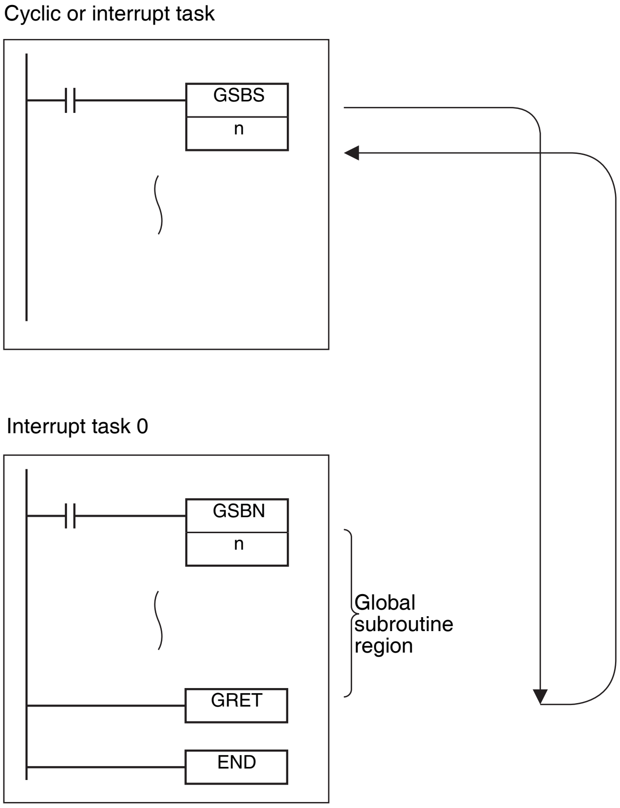
There are no flags affected by this instruction.

Function

● GSBN

GSBN(751) indicates the beginning of the global subroutine with the specified subroutine number.

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).

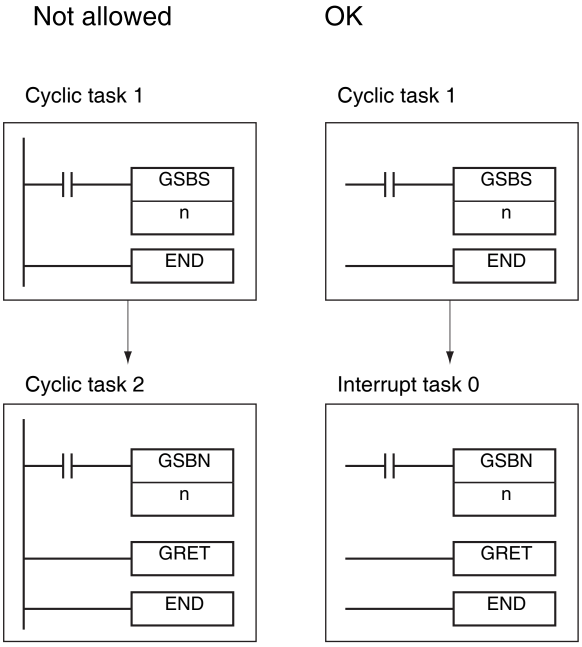


● GRET

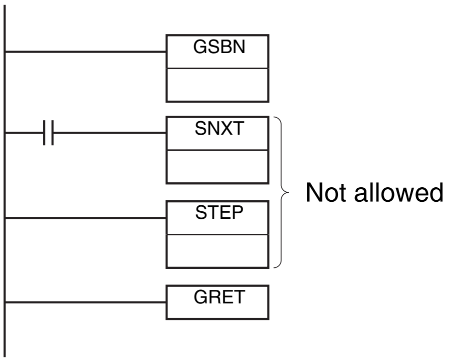
GRET(752) indicates the end of a global subroutine. When program execution reaches GRET(752) it is automatically returned to the next instruction after the GSBS(750) instruction that called the global subroutine.

Precautions

- Always place the global subroutines in interrupt task 0.

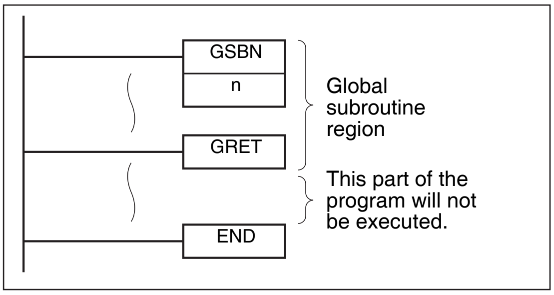


- The step instructions, STEP(008) and SNXT(009) cannot be used in global subroutines.



- 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.

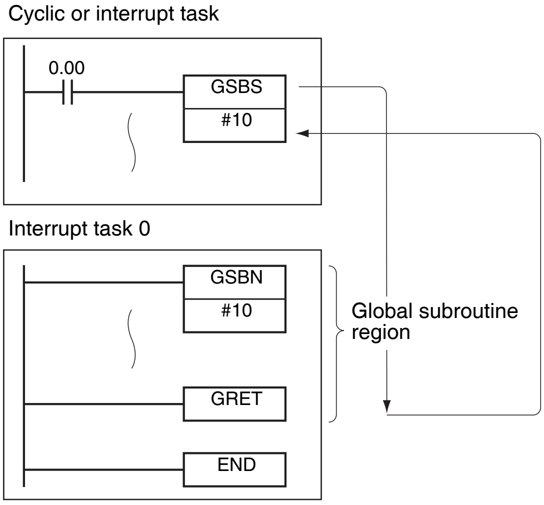
Interrupt task 0



Note : The input method for the global subroutine number, N, is different for the CX-Programmer and a Programming Console. Input #0 to #1023 on the CX-Programmer and 0 to 1023 on a Programming Console.

Example Programming

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.



Interrupt Control Instructions

Interrupt Control Instructions

The CS/CJ-series CPU Units support the following interrupts.

Type	Execution condition	Setting procedure
I/O Interrupts	Interrupt input from the Interrupt Input Unit on the CPU Rack turns ON/OFF.	Use the MSKS instruction to assign inputs from Interrupt Input Units on the CPU Rack.
Scheduled Interrupts	Scheduled (fixed intervals)	Use the MSKS instruction to set the interrupt interval. See Scheduled Interrupt Time Units in PLC Setup.
Power OFF Interrupt	When power turns OFF (After the default power OFF detection time + power OFF detection delay time)	See Power OFF Interrupt Task and Power OFF Detection Delay Time in PLC Setup.
External Interrupts	When requested by an Special I/O Unit or CPU Bus Unit on the CPU Rack or by an Inner Board (CS Series only)	The interrupt conditions and interrupt task numbers are specified in the settings for Special I/O Units, CPU Bus Units, and Inner Boards that support interrupts. MSKS(690) is not used.
Input Interrupts	When interrupt inputs built into the CPU Unit turn ON or OFF (CJ2M-CPU□□ with CJ2M-MD21□ Pulse I/O Module mounted or CJ1M-CPU2□ Only)	Use the MSKS(690) instruction to assign built-in inputs as interrupt inputs.
High-speed Counter Interrupts	When high-speed counter inputs built into the CPU Unit meet conditions for target-value comparison or range comparison (CJ2M-CPU□□ with CJ2M-MD21□ Pulse I/O Module mounted or CJ1M-CPU2□ Only)	Use the CTBL (COMPARISON TABLE LOAD) instruction to specify the target value and interrupt task number. MSKS(690) cannot be used.

Outline of Interrupt Control Instructions

● SET INTERRUPT MASK: MSKS(690)

I/O interrupts, input interrupts, and scheduled interrupts are masked (disabled) when the PLC enters RUN mode. MSKS(690) can be used to unmask or mask interrupt inputs for I/O interrupts/input interrupts and to start/stop the internal timers for scheduled interrupts.

● CLEAR INTERRUPT: CLI(691)

CLI(691) clears or retains recorded interrupt inputs for I/O interrupts/input interrupts or sets the time to the first scheduled interrupt for scheduled interrupts. It also clears or retains causes of high-speed counter interrupts for CJ1M and CJ2M CPU Units.

● READ INTERRUPT MASK: MSKR(692)

MSKR(692) reads the current interrupt processing settings that were set with MSKS(690).

● DISABLE INTERRUPTS: DI(693)

Temporarily disables execution of all interrupt tasks except the power OFF interrupt. All interrupt causes that occur while interrupts are disabled will be recorded.

● ENABLE INTERRUPTS: EI(694)

Enables execution of all interrupt tasks (excluding power OFF interrupt task) that were disabled with DI(693). If interrupt causes were recorded while interrupts were disabled, the corresponding interrupt tasks are executed.

Precautions in Using Interrupt Tasks

● Precautions for All Interrupts

- When IORF(097), FIORF(225), IORD(222), or IOWR(223) 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 enabled in the PLC Setup and one of the following operations occurs during an interrupt task, a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A402.13) will be turned ON.
  - I/O refreshing is performed for the same Special I/O Unit by IORF(097) or FIORF(225).
  - The same Special I/O Unit's data area is read by IORD(222) or written by IOWR(223).
- When using a CS-series PLC, be sure that the interrupt task does not require more than 10 ms if a C200H Special I/O Unit is mounted or SYSMAC BUS Remote I/O Slave Rack is connected. If an interrupt task longer than 10 ms is executed during I/O refreshing with the Special I/O Unit or Slave Rack, a non-fatal will occur and the Interrupt Task Error Flag (A40213) will be turned ON.

● Precautions for I/O Interrupts

- Only interrupt inputs from CS/CJ-series Interrupt Input Units and C200H Interrupt Input Units are supported as causes to trigger I/O interrupt tasks. Interrupt inputs from Inner Boards and Special I/O Units are not supported.
- Interrupt Input Units must be mounted in one of the following slots on the CPU Rack. I/O interrupt tasks will not be executed if the Interrupt Input Unit is mounted in any other slot.
  - CJ2 CPU Units
    - CJ2H-CPU6□-EIP: Slots 0 to 3
    - CJ2H-CPU6□: Slots 0 to 4
  - CJ2M CPU Units: Slots 0 to 4
  - CJ1H CPU Units: Slots 0 to 3
  - CJ1M CPU Units: Slots 0 to 2
- Words are allocated to Interrupt Input Units in the order that they are mounted from left to right.
- The CS1W-INT01 and the C200HS-INT01 cannot be used at the same time.
- All interrupt inputs that have been detected will be cleared when the interrupt mask is cleared.
- Furthermore, the recorded interrupt cause is not cleared until the corresponding interrupt task has been completed, so a new interrupt input will be ignored if it is received while the corresponding interrupt task is being executed.

● Precautions for Scheduled Interrupts

- Be sure that the scheduled interrupt set time interval is longer than the time required to execute the scheduled interrupt task.
- For scheduled interrupts, MSKS(690) is used only to set the scheduled interrupt interval and does not set the time to the first scheduled interrupt. To accurately control the time to the first scheduled interrupt, program CLI(691) to set the time to the first scheduled interrupt just before programming MSKS(690). If MSKS(690) is used to restart a scheduled interrupt for a CJ1M or CJ2M CPU Unit, however, the time to the first scheduled interrupt will be accurate even if CLI(691) is not used.
- The time unit for the scheduled interrupt is set in the PLC Setup as the Scheduled Interrupt Interval.
- Scheduled interrupt tasks cannot be used for CJ2H CPU Units if synchronous unit operation is being used.

Related Memory Area Words

Name	Address	Operation
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 Duplicate Refresh Error Flag	A402.13	ON in the following cases: 1. An interrupt task longer than 10 ms was executed during I/O refreshing with a C200H Special I/O Unit or Remote I/O Slave Rack. (CS Series only) 2. Interrupt Task Error Detection is enabled in the PLC Setup, and one of the following conditions occurs for the same Special I/O Unit. • There is a conflict between an IORF(097), FIORF(225), IORD(222), or IOWR(223) instruction executed in the interrupt task and an IORF(097), FIORF(225), IORD(222), or IOWR(223) instruction executed in the cyclic task. • There is a conflict between an IORF(097), FIORF(225), IORD(222), or IOWR(223) instruction executed in the interrupt task and the CPU Unit's I/O refreshing (END refreshing). **Note** When a Special I/O Unit's Cyclic Refreshing is enabled in the PLC Setup, and an IORF(097), FIORF(225) (CJ1-H-R and CJ2 only), IORD(222), or IOWR(223) instruction is executed for the same Special I/O Unit, there will be duplicate refreshing and an Interrupt Task Error will occur.
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	For error 1: Indicates the interrupt task number. For error 2: Indicates the unit number of the Special I/O Unit where the multiple I/O refreshing occurred.

Related PLC Setup Settings

● Scheduled Interrupts

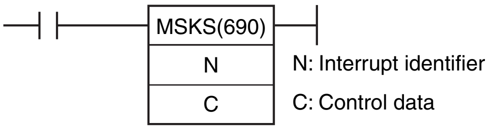
Name	Description	Settings
Scheduled Interrupt Interval	Specify the time unit to use to specify the scheduled interrupt time. The scheduled interrupt time is set using MSKS(690).	10 ms (default) 1.0 ms 0.1 ms

● Power OFF Interrupt

Name	Description	Settings
Power OFF Interrupt Task	If the Power OFF Interrupt Task setting is turned ON, then a power OFF interrupt task will start if power turns OFF.	OFF (default) ON
Power OFF Detection Delay Time	Power OFF is recognized when this time plus the default power OFF detection time (10 to 25 ms for AC power supplies and 2 to 25 ms for DC power supplies) expires.	0 to 10 ms (1-ms units)

MSKS

Instruction	Mnemonic	Variations	Function code	Function
SET INTERRUPT MASK	MSKS	@MSKS	690	Controls interrupts.

	MSKS
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Interrupt identifier	---	1
C	Control data	UINT	1

(1) Disabling/Enabling Input Interrupts for I/O Interrupt Tasks

Operand	Contents	
N	**Interrupt input**	**Interrupt identifier**
	Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	0: Unit number 0 I/O interrupts 0 to 15 (interrupt tasks 100 to 115) 1: Unit number 1 I/O interrupts 16 to 31 (interrupt tasks 116 to 131)
	Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)	0: Unit number 0 I/O interrupts 0 to 7 (interrupt tasks 100 to 107) 1: Unit number 1 I/O interrupts 8 to 15 (interrupt tasks 108 to 115) 2: Unit number 2 I/O interrupts 16 to 23 (interrupt tasks 116 to 123) 3: Unit number 3 I/O interrupts 24 to 31 (interrupt tasks 124 to 131)
	Inputs to a CJ1M CPU Unit's Built-in Inputs (4 inputs/Unit)	6: Interrupt input 0 (interrupt task 140) 7: Interrupt input 1 (interrupt task 141) 8: Interrupt input 2 (interrupt task 142) 9: Interrupt input 3 (interrupt task 143)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	100: Input interrupt 0 (Block 0) (interrupt task 140) 101: Input interrupt 1 (Block 0) (interrupt task 141) 102: Input interrupt 2 (Block 0) (interrupt task 142) 103: Input interrupt 3 (Block 0) (interrupt task 143) 104: Input interrupt 4 (Block 1) (interrupt task 144) 105: Input interrupt 5 (Block 1) (interrupt task 145) 106: Input interrupt 6 (Block 1) (interrupt task 146) 107: Input interrupt 7 (Block 1) (interrupt task 147) **Note** For input interrupts 0 to 3, 6 to 9 can also be used as the interrupt number.
C	**Interrupt input**	**Interrupt mask**
	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	Set to 0000 to FFFF hex Bits 0 to 15 correspond to each interrupt task. Individual bit settings are as follows: 0: Enable (unmask) the interrupt. 1: Disable (mask) the interrupt.
	Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)	Set to 0000 to 00FF hex Bits 0 to 7 correspond to each interrupt task. Individual bit settings are as follows: 0: Enable (unmasks) the interrupt. 1: Disable (mask) the interrupt.
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit) CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	0000 hex: Enable (unmask) the interrupt (direct mode). 0001 hex: Disable (mask) the interrupt (direct mode). 0002 hex: Start decrementing counter and enable interrupt (counter mode). 0003 hex: Start incrementing counter and enable interrupt (counter mode).

(2) Specifying Up/Down Differentiation of Interrupt Inputs for I/O Interrupts and Input Interrupts (Except when Using a C200HS-INT01 Interrupt Input Unit)

Operand	Contents	
N	**Interrupt input**	**Interrupt identifier**
	Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	2: Unit number 0 I/O interrupts 0 to 15 (interrupt tasks 100 to 115) 3: Unit number 1 I/O interrupts 16 to 31 (interrupt tasks 116 to 131)
	Inputs to a CJ1M CPU Unit's Built-in Inputs (4 inputs/Unit)	10: Interrupt input 0 (interrupt task 140) 11: Interrupt input 1 (interrupt task 141) 12: Interrupt input 2 (interrupt task 142) 13: Interrupt input 3 (interrupt task 143)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	110: Input interrupt 0 (Block 0) (interrupt task 140) 111: Input interrupt 1 (Block 0) (interrupt task 141) 112: Input interrupt 2 (Block 0) (interrupt task 142) 113: Input interrupt 3 (Block 0) (interrupt task 143) 114: Input interrupt 4 (Block 1) (interrupt task 144) 115: Input interrupt 5 (Block 1) (interrupt task 145) 116: Input interrupt 6 (Block 1) (interrupt task 146) 117: Input interrupt 7 (Block 1) (interrupt task 147) **Note** For input interrupts 0 to 3, 10 to 13 can also be used as the interrupt number.
C	**Interrupt input**	**Up/down differentiation specification**
	Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	Set to 0000 to FFFF hex. Bits 0 to 15 correspond to each interrupt task. Individual bit settings are as follows: 0: Up-differentiation (Detect rising edge.) 1: Down-differentiation (Detect falling edge.)
	Inputs to a CJ1M CPU Unit's Built-in Inputs (4 inputs/Unit)	0000 hex: Up-differentiation (Detect rising edge.) 0001 hex: Down-differentiation (Detect falling edge.)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	

Note When the up/down differentiation setting is changed, all detected interrupt inputs will be cleared.

(3) Disabling/Enabling a Scheduled Interrupt Task's Timer Interrupt

Operand	Contents	
N	**Scheduled Interrupt No.**	
	4: Interrupt task 0 (interrupt task 2) 5: Interrupt task 1 (interrupt task 3) **Note 1** Only scheduled interrupt 0 can be used with the CJ1M-CPU11/21. **2** The scheduled interrupts (0 and 1) cannot be used with a CJ2H CPU Unit if the synchronous unit operation function is enabled.	
C	**Scheduled interrupt time units (Set in the PLC Setup.)**	**Scheduled interrupt set time**
	Any time unit setting	0 decimal (0000 hex): Disable interrupt. (Stop internal timer.)
	10 ms	1 to 9,999 decimal (0001 to 270F hex): Enable interrupt. (Start internal timer with interrupt interval between 10 and 99,990 ms.)
	1 ms	1 to 9,999 decimal (0001 to 270F hex): Enable interrupt. (Start internal timer with interrupt interval between 1 and 9,999 ms.)
	0.1 ms	CJ1M CPU Units 5 to 9,999 decimal (0005 to 270F hex): Enable interrupt. (Start internal timer with interrupt interval between 0.5 and 999.9 ms.) **Note** Settings 0001 to 0004 cannot be used. An error will occur if one of these settings is used.
		CJ1-H-R and CJ2 CPU Units 2 to 9,999 decimal (0002 to 270F hex): Enable interrupt. (Start internal timer with interrupt interval between 0.2 and 999.9 ms.) **Note** Setting 0001 cannot be used. An error will occur if 0001 is set. (See note.) CJ2M CPU Units 4 to 9,999 decimal (0004 to 270F hex): Enable interrupt. (Start internal timer with interrupt interval between 0.4 and 999.9 ms.) **Note** Settings 0001 to 0003 decimal (0001 to 0003 hex) cannot be used. An error will occur if one of these settings is used.

Note High-speed interrupt function can be used with CJ2 CPU Units with unit version 1.1 or later to set an interrupt interval of 0.1 ms (&1 decimal or #0001 hex) for scheduled interrupt 1 (interrupt task 2). This setting cannot be used for other interrupts.

(4) Resetting and Starting Scheduled Interrupts (CJ1M and CJ2M CPU Units Only)

Operand	Contents	
N	**Scheduled Interrupt No.**	
	14: Scheduled interrupt 0 (interrupt task 2) 15: Scheduled interrupt 1 (interrupt task 3) **Note** Only scheduled interrupt 0 can be used with the CJ1M-CPU11/21.	
C	**Scheduled interrupt time units (Set in the PLC Setup.)**	**Scheduled interrupt set time**
	Any time unit setting	0 decimal (0000 hex): Disable interrupt. (Stop internal timer.)
	10 ms	1 to 9,999 decimal (0001 to 270F hex): Enable interrupt. (Reset internal timer value, and then start the timer with an interrupt interval between 10 and 99,990 ms.)
	1 ms	1 to 9,999 decimal (0001 to 270F hex): Enable interrupt. (Reset internal timer value, and then start timer with an interrupt interval between 1 and 9,999 ms.)
	0.1 ms	5 to 9,999 decimal (0005 to 270F hex): Enable interrupt. (Reset internal timer value, and then start timer with interrupt interval between 0.5 and 999.9 ms.) **Note** Settings 0001 to 0004 cannot be used. An error will occur if one of these settings is used.
		CJ2M CPU Units 4 to 9,999 decimal (0004 to 270F hex): Enable interrupt. (Reset internal timer and then start internal timer with interrupt interval between 0.4 and 999.9 ms.) **Note** Settings 0001 to 0003 decimal (0001 to 0003 hex) cannot be used. An error will occur if one of these settings is used.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N is not within the specified range (0 to 15 for the CJ1M CPU Unit, 0 to 15, 100 to 107, or 110 to 117 for the CJ2M CPU Unit; and 0 to 5 for other CPU Units). Errors when specifying I/O Interrupts: • I/O interrupt specified: When using C200HS-INT01 interrupt inputs, the Error Flag will go ON if C is not between 0000 and 00FF hex. • I/O interrupt specified: When using the CJ1M or CJ2M CPU Unit's built-in interrupt inputs, the Error Flag will go ON if C is not between 0 and 3. Errors when specifying Scheduled Interrupts: • I/O interrupt specified: For a CJ2M CPU Unit, ON for any function that uses inputs on the Interrupt Input Unit even if a CJ1W-INT01 Interrupt Input Unit is not mounted. • I/O interrupt specified: ON if using input interrupts is not enabled for the port specified with N when enabling/disabling the input interrupt or changing the differentiation specification (edge) for a CJ1M/CJ2M CPU Unit built-in input. • I/O interrupt specified: ON if using interrupts is enabled/disabled or the differentiation specification (edge) is changed while executing interrupt feeding (IFEED(892)) using the specified interrupt input for a CJ2M CPU Unit built-in interrupt input. • Scheduled interrupt specified: When the time units are set to 10 ms or 1 ms, the Error Flag will go ON if C is not between 0 and 9,999 decimal (0000 to 270F hex). • Scheduled interrupt specified: When using a CJ1M CPU Unit with the time units set to 0.1 ms, the Error Flag will go ON if C is not between 5 and 9,999 decimal (0005 to 270F hex). • Scheduled interrupt specified: When using a CJ1-H-R or CJ2H CPU Unit with the time units set to 0.1 ms, the Error Flag will go ON if C is not 0 or between 2 and 9,999 decimal (0000 or 0002 to 270F hex). (See note.) • Scheduled interrupt specified: When using a CJ2M CPU Unit with the time unit set to 0.1 ms, the Error Flag will go ON if C is not 0 or between 4 and 9,999 decimal (0000 or 0004 to 270F hex). • ON if N is set to 4 or 5 (scheduled interrupt designation) for a CJ2H CPU Unit and the synchronous unit operation function is enabled. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.
Equals Flag	P_EQ	OFF
Negative Flag	P_N	OFF

Note If high-speed interrupt function is enabled for a CJ2 CPU Unit with unit version 1.1 or later, an error will not occur and the Error Flag will remain OFF even if the time unit is set to 0.1 ms for scheduled interrupt 0 (N = 4).

Related PLC Setup Settings

Name	Description	Settings
Scheduled Interrupt Interval	Specifies the time unit to use to specify the scheduled interrupt time. Set the time unit when executing scheduled interrupts. The scheduled interrupt time is set using MSKS(690).	10 ms (default) 1.0 ms 0.1 ms

Related Auxiliary Area Flags and Words

Name	Address	Operation
Duplicate Refresh Error Flag	A402.13	ON in the following cases: 1. If Interrupt Task Error Detection is enabled in the PLC Setup, the Interrupt Task Error Flag will turn ON if the following conditions occur for the same Special I/O Unit. •There is a conflict between an IORF, FIORF, IORD, or IOWR instruction executed in the interrupt task and an IORF, FIORF, IORD, or IOWR instruction executed in the cyclic task. •There is a conflict between an IORF, FIORF, IORD, or IOWR instruction executed in the interrupt task and the CPU Unit's I/O refreshing (END refreshing). 2. An interrupt task longer than 10 ms was executed during I/O refreshing with a C200H Special I/O Unit or Remote I/O Slave Rack. (CS Series only)
Duplicate Refresh Error Cause Flag	A426.15	This stores the cause when A402.13 (Duplicate Refresh Error Flag) is ON. 1: Duplicate refresh 0: Execute interrupt task for at least 10 ms during refresh of I/O with a C200H Special I/O Unit or SYSMAC BUS Remote I/O Slave Rack (CS Series only)
Duplicate Refresh Error Unit Number	A426.00 to A426.11	When A402.13 (Duplicate Refresh Error Flag) is ON, the information below is stored as 12 bits of binary data depending on the status of A426.15 (Duplicate Refresh Cause Flag). • When A426.15 is 1 (ON), the unit number of the duplicate-refreshed special I/O unit • When A426.15 is 0 (OFF), the task number of the interrupt task executed for 10 ms or more (CS Series only)

Function

When the program execution starts, the interrupt inputs that generate I/O interrupts/input interrupts are masked (disabled), and the internal timers creating the timer interrupts that generate scheduled interrupt tasks are stopped.

Use MSKS(690) to enable the I/O interrupts/input interrupts and timer interrupts, so that the corresponding interrupt tasks can be executed.

MSKS(690) controls the execution of interrupt tasks. The value of N specifies the interrupt task and the kind of processing that will be performed.

(1) Enabling/Disabling the I/O Interrupts and Input Interrupts (N = 0 to 3, 6 to 9, or 100 to 107)

- Enables or disables the interrupt inputs specified by N, based on the status of the bits in C. With this function, MSKS(690) can control whether or not each I/O interrupt task and input interrupt task is executed.
- When an interrupt input is enabled, any interrupts detected up to that point will be cleared.

(2) Specifying the Differentiation for I/O Interrupts and Input Interrupts (N = 2, 3, 10 to 13, or 110 to 117)

- Specifies whether the interrupt inputs specified by N are up-differentiated or down-differentiated, based on the status of the bits in C.
- Use the differentiation specification together with the enabling/disabling function. If MSKS(690) is not executed to specify up or down differentiation, the interrupt inputs are up-differentiated (the default setting).
- When MSKS(690) is executed to specify an interrupt input's up or down differentiation, any interrupts detected up to that point will be cleared.

(3) Starting and Stopping Internal Timers for Scheduled Interrupts (N = 4 or 5)

- Sets the scheduled interrupt interval (specified by C) for the specified scheduled interrupt (specified by N) and starts the internal timer. The internal timer can also be stopped. With this function, MSKS(690) can control whether or not each scheduled task is executed.

- When MSKS(690) is used to restart the internal timer, the time from the execution of MSKS(690) to the start of the first scheduled interrupt task is uncertain, because the existing internal timer PV is used.
  When you want to specify the interrupt start time, use CLI(691) together with MSKS(690).

(4) Resetting Internal Timers for Scheduled Interrupts (N = 14 or 15)

- Sets the time interval (specified by C) for the specified scheduled interrupt task (specified by N), resets the internal timer's PV, and starts the internal timer. Since the internal timer's PV is reset, this function maintains the proper interval from the execution of MSKS(690) until the start of the first interrupt. (This operation is different from item (3), above.) (CJ1M and CJ2M CPU Units Only)

Note 1 The CJ1M-CPU11/21 supports only one scheduled interrupt task, interrupt task 2 for scheduled interrupt 0.

2 The time unit for the scheduled interrupt time is set as the Schedule Interrupt Interval in the PLC Setup (default: 10 ms).

Hint

The longest interrupt task processing time is stored in A440 (Maximum Interrupt Task Processing Time). At the same time, the task number of the interrupt task with the longest interrupt task processing time is stored in A441 (Interrupt Task with Maximum Processing Time).

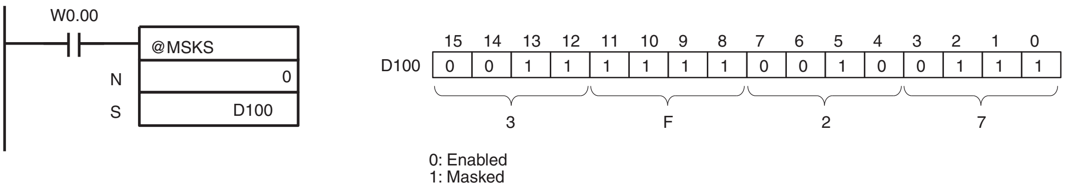
Precaution

- Be sure that the scheduled interrupt set time interval is longer than the time required to execute the scheduled interrupt task.
- For scheduled interrupts, MSKS(690) is used only to set the scheduled interrupt interval and does not set the time to the first scheduled interrupt after execution of MSKS(690). The internal timer when MSKS(690) is executed will continue from the value where is previously stopped. This means that the time to the first scheduled interrupt will not be consistent. To accurately control the time to the first interrupt and the interrupt interval, program CLI(691) to set the time to the first scheduled interrupt just before programming MSKS(690). If MSKS(690) is used to restart a scheduled interrupt for a CJ1M or CJ2M CPU Unit, however, the time to the first scheduled interrupt will be accurate even if CLI(691) is not used.
- Scheduled interrupt tasks cannot be used for CJ2H CPU Units if synchronous unit operation is being used. An instruction error will occur if the MSKS(690) instruction is executed with N (interrupt identifier) set to 4 or 5.
- Since the execution of an interrupt task is prohibited during the execution of the MSKS instruction, the execution of the interrupt task may be delayed, or the interrupt task may not be executed.

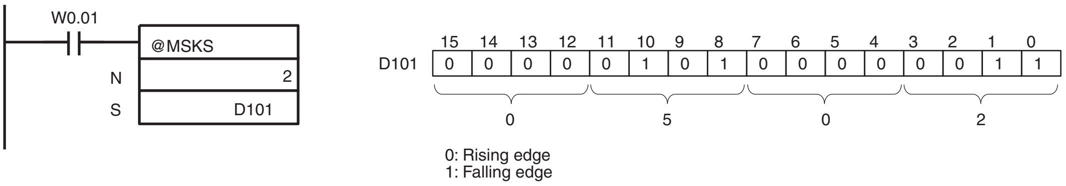
Example Programming

● Examples for CS1W-INT01/CJ1W-INT01

When W0.00 goes from OFF to ON, MSKS(690) unmasks (enables) interrupt inputs in Interrupt Input Unit 0.

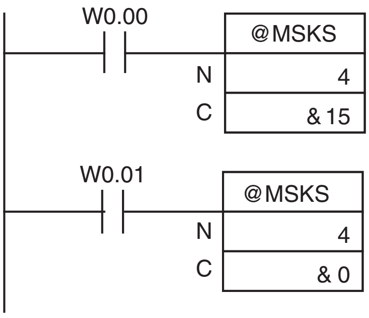


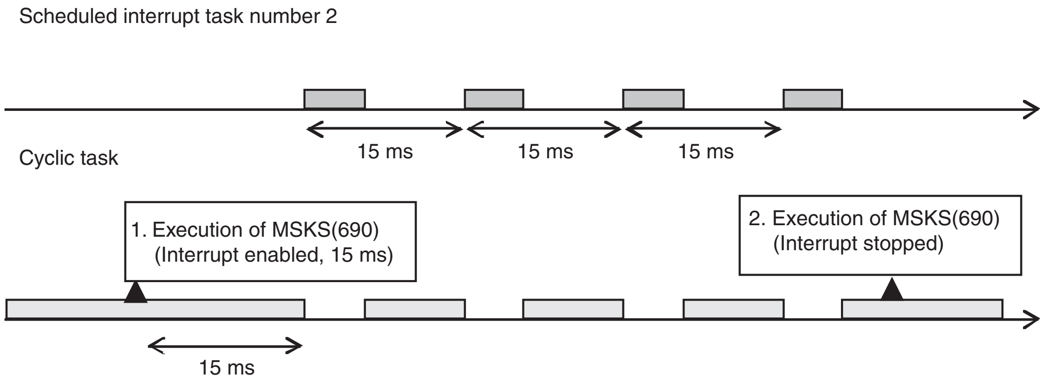
When W0.01 goes from OFF to ON, MSKS(690) sets the rising/falling edge designations for Interrupt Input Unit 0.



Example for Scheduled Interrupts

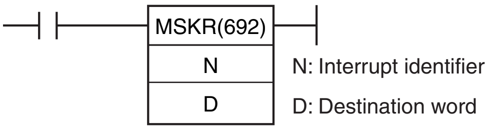
1. When W0.00 goes from OFF to ON in the following example, MSKS(690) sets a 15 ms time interval for scheduled interrupt 0, and starts the internal timer. (In this case, the scheduled time interval units are set to 1 ms.)
2. When W0.01 goes from OFF to ON, the internal timer is stopped for scheduled interrupt 0, which stops the generation of timer interrupts.





MSKR

Instruction	Mnemonic	Variations	Function code	Function
READ INTERRUPT MASK	MSKR	@MSKR	692	Reads the current interrupt control settings that were set with MSKS(690).

Symbol	**MSKR**
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Interrupt identifier	---	1
D	Destination word	UINT	1

(1) Reading the Interrupt Mask Settings Set for I/O Interrupts and Input Interrupts

Operand	Contents	
	Interrupt input	**Interrupt identifier**
N	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	0: Interrupt Input Unit 0 I/O interrupts 0 to 15 (interrupt tasks 100 to 115) 1: Interrupt Input Unit 1 I/O interrupts 16 to 31 (interrupt tasks 116 to 131)
	Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)	0: Interrupt Input Unit 0 I/O interrupts 0 to 7 (interrupt tasks 100 to 107) 1: Interrupt Input Unit 1 I/O interrupts 8 to 15 (interrupt tasks 108 to 115) 2: Interrupt Input Unit 2 I/O interrupts 16 to 23 (interrupt tasks 116 to 123) 3: Interrupt Input Unit 3 I/O interrupts 24 to 31 (interrupt tasks 124 to 131)
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit)	6: Input interrupt 0 (interrupt task 140) 7: Input interrupt 1 (interrupt task 141) 8: Input interrupt 2 (interrupt task 142) 9: Input interrupt 3 (interrupt task 143)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	100: Input interrupt 0 (interrupt task 140) 101: Input interrupt 1 (interrupt task 141) 102: Input interrupt 2 (interrupt task 142) 103: 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) **Note** For input interrupts 0 to 3, 6 to 9 can also be used as the interrupt number.

Operand	Contents	
D	**Interrupt input**	**Read interrupt mask status**
	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	Range: 0000 to FFFF hex Bits 0 to 15 correspond to each interrupt task. The meaning of the individual flags is as follows: 1: Disable (mask) the interrupt. 0: Enable (unmask) the interrupt.
	Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)	Range: 0000 to 00FF hex Bits 0 to 7 correspond to each interrupt task. The meaning of the individual flags is as follows: 1: Disable (mask) the interrupt. 0: Enable (unmask) the interrupt.
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit)	0000 hex: Interrupts enabled (unmasked) in direct mode. 0001 hex: Interrupts disabled (masked) in direct mode. 0002 hex: Interrupts enabled for decrementing counter in counter mode. 0003 hex: Interrupts enabled for incrementing counter in counter mode.
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	

(2) Reading the Differentiation Settings Set for I/O Interrupts and Input Interrupts (Except when Using a C200HS-INT01 Interrupt Input Unit)

Operand	Contents	
N	**Interrupt input**	**Interrupt identifier**
	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	2: Unit number 0 (interrupt tasks 100 to 115) I/O interrupts 0 to 15 (interrupt tasks 100 to 115) 3: Unit number 1 (interrupt tasks 116 to 131) I/O interrupts 16 to 31 (interrupt tasks 116 to 131)
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit)	10: Interrupt input 0 (interrupt task 140) 11: Interrupt input 1 (interrupt task 141) 12: Interrupt input 2 (interrupt task 142) 13: Interrupt input 3 (interrupt task 143)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	110: Input interrupt 0 (Block 0) (interrupt task 140) 111: Input interrupt 1 (Block 0) (interrupt task 141) 112: Input interrupt 2 (Block 0) (interrupt task 142) 113: Input interrupt 3 (Block 0) (interrupt task 143) 114: Input interrupt 4 (Block 1) (interrupt task 144) 115: Input interrupt 5 (Block 1) (interrupt task 145) 116: Input interrupt 6 (Block 1) (interrupt task 146) 117: Input interrupt 7 (Block 1) (interrupt task 147) **Note** For input interrupts 0 to 3, 10 to 13 can also be used as the interrupt number.
D	**Interrupt Input**	**Read interrupt input differentiation setting**
	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	Range: 0000 to FFFF hex. Bits 0 to 15 correspond to each interrupt task. The meaning of the individual flags is as follows: 0: Up-differentiation (Detect rising edge.) 1: Down-differentiation (Detect falling edge.)
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit)	0000 hex: Up-differentiation (Detect rising edge.) 0001 hex: Down-differentiation (Detect falling edge.)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	

(3) Reading the Set Value of a Scheduled Interrupt Task's Internal Timer

Operand	Contents	
N	**Scheduled Interrupt No.**	
	4: Interrupt task 0 (interrupt task 2) 5: Interrupt task 1 (interrupt task 3) **Note** Only scheduled interrupt 0 can be used with the CJ1M-CPU11/21.	
D	**Scheduled interrupt time units (Set in the PLC Setup.)**	**Scheduled interrupt set time**
	Any time unit setting	0 decimal (0000 hex): Interrupt disabled. (Internal timer stopped.)
	10 ms	1 to 9,999 decimal (0001 to 270F hex): Interrupt enabled. (Internal timer started with interrupt interval between 10 and 99,990 ms.)
	1 ms	1 to 9,999 decimal (0001 to 270F hex): Interrupt enabled. (Internal timer started with interrupt interval between 1 and 9,999 ms.)
	0.1 ms	1 to 9,999 decimal (0001 to 270F hex): Interrupt enabled. (Internal timer started with interrupt interval between 0.1 and 999.9 ms.)

(4) Reading the Present Value of a Scheduled Interrupts Internal Timer (CJ1M and CJ2M CPU Units Only)

Operand	Contents	
N	**Interrupt identifier**	
	14: Scheduled interrupt 0 (interrupt task 2) 15: Scheduled interrupt 1 (interrupt task 3) **Note** Only scheduled interrupt 0 can be used with the CJ1M-CPU11/21.	
D	**Scheduled interrupt time units (Set in the PLC Setup.)**	**Read internal timer PV**
	10 ms	0 to 9,999 decimal (0000 to 270F hex): Interrupt timer PV between 0 and 99,990 ms
	1 ms	0 to 9,999 decimal (0000 to 270F hex): Interrupt timer PV between 1 and 9,999 ms.
	0.1 ms	0 to 9,999 decimal (0000 to 270F hex): Interrupt timer PV between 0.0 and 999.9 ms.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if N is not within the specified range (0 to 15 for the CJ1M CPU Unit, 0 to 15, 100 to 107, or 110 to 117 for the CJ2M CPU Unit; and 0 to 5 for other CPU Units). • I/O interrupt specified: For a CJ2M CPU Unit, ON for any function that uses inputs on the Interrupt Input Unit even if a CJ1W-INT01 Interrupt Input Unit is not mounted. • I/O interrupt specified: ON if using input interrupts is not enabled for the port specified with N when the mask status or differentiation (edge) specification is read for a CJ1M/CJ2M CPU Unit built-in input. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

MSKR(692) reads the interrupt task settings that were set with MSKS(690). The value of N specifies the interrupt task and the kind of information that will be read.

(1) Interrupt Mask Status for I/O Interrupts and Input Interrupts (N = 0 to 3, 6 to 9, or 100 to 107)

Reads the masked/unmasked status of the input interrupts specified by N, and outputs that information to the bits in D.

(2) Specifying the Interrupt Input Differentiation for I/O Interrupts and Input Interrupts (N = 2, 3, 10 to 13, or 110 to 117)

Reads the up/down differentiation settings of the interrupt inputs specified by N, and outputs that information to the bits in D.

(3) Setting Internal Timers for Scheduled Interrupts (N = 4 or 5)

Reads the time set for the internal timer of the scheduled interrupt specified by N, and outputs that information to D.

(4) N = 14 or 15: Reading a Scheduled Interrupt's Internal Timer PV

Reads the internal timer PV of the scheduled interrupt specified by N, and outputs that information to D. The internal timer's PV is the time that has elapsed since the scheduled interrupt started (when MSKS(690) was executed), or the time that has elapsed since the last scheduled interrupt started. (CJ1M and CJ2M CPU Units Only)

Note 1 The CJ1M-CPU11/21 supports only one scheduled interrupt task, interrupt task 2 for scheduled interrupt 0.

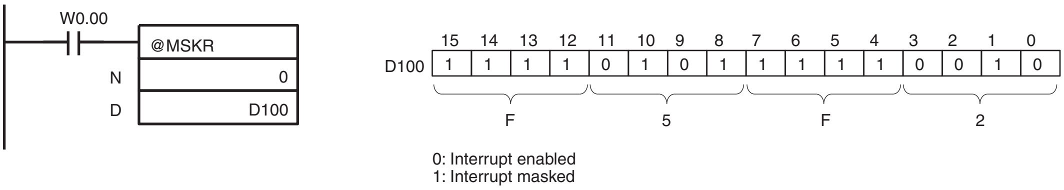
2 The time unit for the scheduled interrupt set time and PV is set as the Schedule Interrupt Interval in the PLC Setup (default:10 ms).

3 MSKR(692) can be executed in the main program or in interrupt tasks.

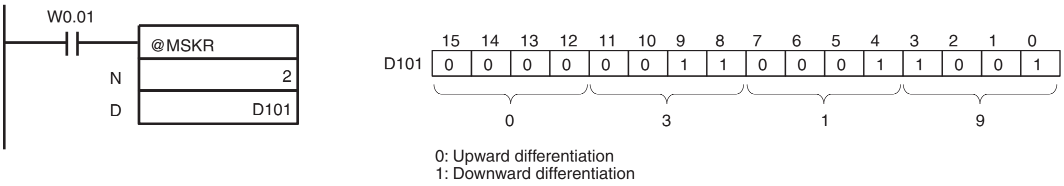
Example Programming

● Specifying I/O Interrupts for the CS1W-INT01 or CJ1W-INT01

When W0.00 goes from OFF to ON, MSKR(692) reads the current mask status of Interrupt Input Unit 0 and stores it in D100.

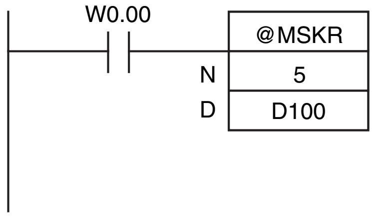


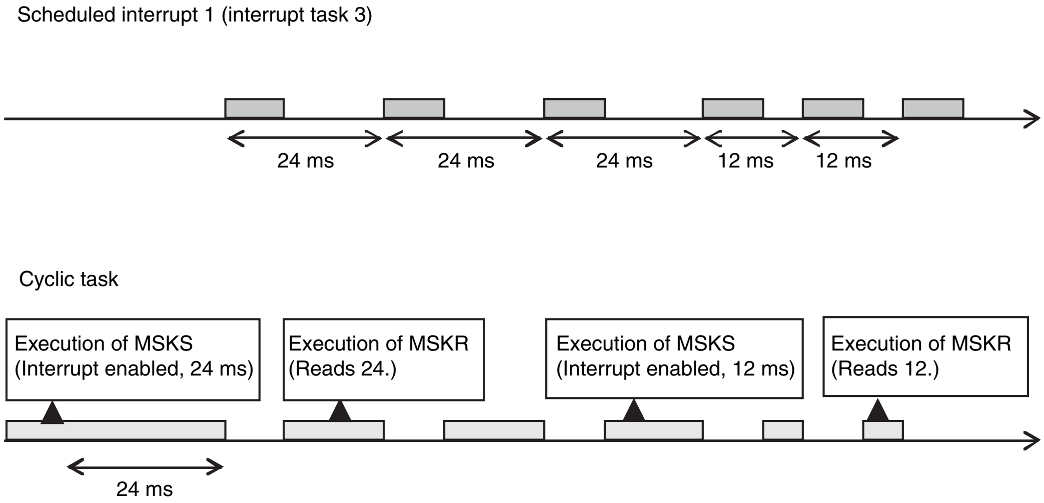
When W0.01 goes from OFF to ON, MSKS(690) reads the upward/downward differentiation for Interrupt Input Unit 0 and stores it in D101.



Example for Scheduled Interrupts

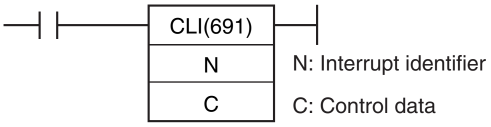
When W0.00 goes from OFF to ON while the internal timer is operating for scheduled interrupt 1, MSKR(692) reads the interrupt time interval setting and outputs the setting to D100.





CLI

Instruction	Mnemonic	Variations	Function code	Function
CLEAR INTERRUPT	CLI	@CLI	691	Clears/retains recorded interrupt inputs, sets the time to the first scheduled interrupt for scheduled interrupt tasks.

	CLI
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Interrupt number	---	1
C	Control data	UINT	1

(1) Clearing/Retaining Recorded Interrupt Causes for I/O Interrupts or Input Interrupts

Operand	Contents	
N	**Interrupt input**	**Interrupt identifier**
	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	0: Unit number 0 I/O interrupts 0 to 15 (interrupt tasks 100 to 115) 1: Unit number 1 I/O interrupts 16 to 31 (interrupt tasks 116 to 131)
	Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)	0: Unit number 0 I/O interrupts 0 to 7 (interrupt tasks 100 to 107) 1: Unit number 1 I/O interrupts 8 to 15 (interrupt tasks 108 to 115) 2: Unit number 2 I/O interrupts 16 to 23 (interrupt tasks 116 to 123) 3: Unit number 3 I/O interrupts 24 to 31 (interrupt tasks 124 to 131)
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit)	6: Interrupt input 0 (interrupt task 140) 7: Interrupt input 1 (interrupt task 141) 8: Interrupt input 2 (interrupt task 142) 9: Interrupt input 3 (interrupt task 143)
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	100: Input interrupt 0 (interrupt task 140) 101: Input interrupt 1 (interrupt task 141) 102: Input interrupt 2 (interrupt task 142) 103: 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) **Note** For input interrupts 0 to 3, 6 to 9 can also be used as the interrupt number.

Operand	Contents	
C	**Interrupt input**	**Control data (cause processing data)**
	Inputs to a CS1W-INT01 or CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)	Set to 0000 to FFFF hex. Bits 0 to 15 correspond to each interrupt task. Individual bit settings are as follows: 0: Retain the recorded interrupt. 1: Clear the recorded interrupt.
	Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)	Set to 0000 to 00FF hex Bits 0 to 7 correspond to each interrupt task. Individual bit settings are as follows: 0: Retain the recorded interrupt. 1: Clear the recorded interrupt.
	Inputs to a CJ1M-CPU2□ CPU Unit's Built-in Inputs (4 inputs/Unit)	0001 hex: Clear the recorded interrupt. 0000 hex: Retain the recorded interrupt.
	CJ2M CPU Unit inputs (when Pulse I/O Module is mounted) (4 inputs/Block)	

(2) Setting the Time to the First Scheduled Interrupts

Operand	Contents	
N	**Interrupt identifier**	
	4: Interrupt task 0 (interrupt task 2) 5: Interrupt task 1 (interrupt task 3) **Note 1** Only scheduled interrupt 0 can be used with the CJ1M-CPU11/21. **2** The scheduled interrupts (0 and 1) cannot be used with a CJ2 CPU Unit if the synchronous unit operation function is enabled.	
C	**Scheduled interrupt time units (Set in the PLC Setup.)**	**Scheduled interrupt set time**
	10 ms	0 to 9,999 decimal (0000 to 270F hex): Sets time to first interrupt between 10 and 99,990 ms.
	1 ms	0 to 9,999 decimal (0000 to 270F hex): Sets time to first interrupt between 1 and 9,999 ms.)
	0.1 ms	0 to 9,999 decimal (0000 to 270F hex): Sets time to first interrupt between 0.1 and 999.9 ms.)

(3) Clearing/Retaining High-speed Counter Interrupt Causes (CJ1M or CJ2M CPU Units Only)

Operand	Contents
N	**Interrupt identifier**
	10: High-speed counter input 0 11: High-speed counter input 1 12: High-speed counter input 2 13: High-speed counter input 3
C	**Recorded Interrupt**
	0001 hex: Clear the recorded interrupt. 0000 hex: Retain the recorded interrupt.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK		OK				

Flags

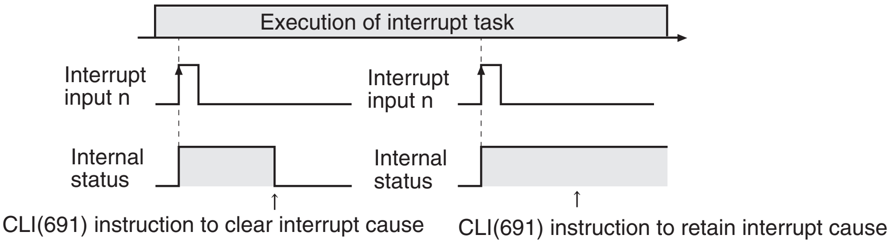
Name	Label	Operation
Error Flag	P_ER	• ON if N is not within the specified range (0, 1, or 4 to 11 for CJ1M CPU Unit; 0, 1, 4 to 13, or 100 to 107 for CJ2M CPU Unit; or 0 to 5 for other CPU Units). • ON if C is not within the specified range of 0000 to 00FF hex when N is 0 to 3 (for I/O interrupts and C200HS-INT only). • ON if C is not 0 or 1 (for I/O interrupts and CJ1M/CJ2M CPU Unit built-in input interrupt or high-speed counter interrupt). • ON if C is not within the specified range of 0 to 9,999 decimal (0000 to 270F hex) for scheduled interrupts. • I/O interrupt specified: For a CJ2M CPU Unit, ON for any function that uses inputs on the Interrupt Input Unit even if a CJ1W-INT01 Interrupt Input Unit is not mounted. • I/O interrupt specified: ON if using input interrupts is not enabled for the port specified with N when input interrupts are cleared or saved for a CJ1M/CJ2M CPU Unit built-in input. • High-speed counter specified: ON if using a high-speed counter is not enabled for the port specified with N when input interrupts are cleared or saved for a CJ1M/CJ2M CPU Unit built-in high-speed counter. • ON when 4 or 5 (scheduled interrupt specification) is set for N when synchronous unit operation is enabled for a CJ2H CPU Unit. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

Depending on the value of N, CLI(691) clears/retains the specified recorded causes of I/O interrupts or input interrupts, sets the time before execution of the first scheduled interrupt, or clears/retains the specified recorded causes of high-speed counter interrupts (CJ1M or CJ2M CPU Unit built-in high-speed counter inputs only).

(1) Clearing/Retaining Recorded Interrupt Causes for I/O Interrupts or Input Interrupts (N = 0 to 3, 6 to 9, or 100 to 107)

CLI(691) clears a recorded cause of the interrupt input specified by N when the corresponding bit of C is ON, and retains the recorded cause of the interrupt input when the corresponding bit is OFF.

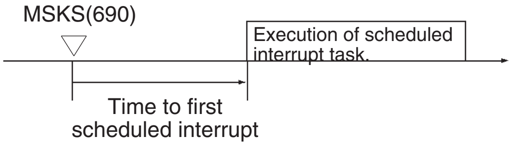


If an interrupt task is being executed and an interrupt input with a different interrupt number is received, execution of the corresponding interrupt task is placed on hold and that interrupt cause is recorded internally. The interrupt tasks for the recorded I/O interrupt causes are executed after execution of the current interrupt task has been completed in order of their priority.

If you want to cancel execution of I/O interrupt tasks or input interrupt tasks that are on hold, use CLI(691) to clear the recorded interrupt causes before the interrupt tasks are executed. Interrupt causes are always recorded and held. No settings are required to record and hold interrupt causes.

(2) Setting the Time to the First Scheduled Interrupt (N = 4 or 5)

When N is 4 or 5, the contents of C specifies the time interval to the first scheduled interrupt.



Note 1 The CJ1M-CPU11/21 supports only one scheduled interrupt task for scheduled interrupt 0 (interrupt task 2).

2 The time unit for the time to the first interrupt is set as the Schedule Interrupt Interval in the PLC Setup (default: 10 ms).

(3) Clearing or Holding High-speed Counter Interrupt Causes (N = 10 or 13)

When N is 10 or 13, CLI(691) clears or retains the recorded high-speed counter interrupt cause (either target-value or range comparison) for the built-in high-speed counter interrupt specified by N. (CJ1M and CJ2M CPU Units Only)

Note CJ1M CPU Units support only high-speed counter inputs 0 and 1.

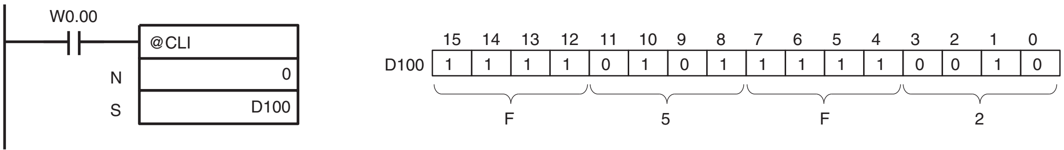
Precautions

- Scheduled interrupt tasks cannot be used for CJ2H CPU Units if synchronous unit operation is being used. An instruction error will occur if the CLI instruction is executed with N set to 4 or 5.
- Since the execution of an interrupt task is prohibited during the execution of the CLI instruction, the execution of the interrupt task may be delayed, or the interrupt task may not be executed.

Example Programming

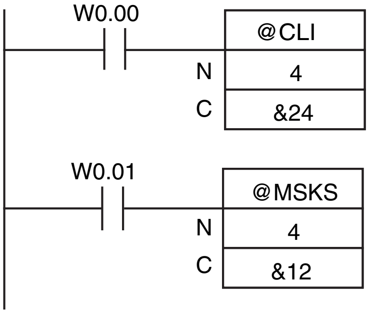
● Specifying I/O Interrupts for the CS1W-INT01 or CJ1W-INT01

When W0.00 goes from OFF to ON, CLI(691) clears or holds the recorded causes for the interrupt inputs from Interrupt Input Unit 0.



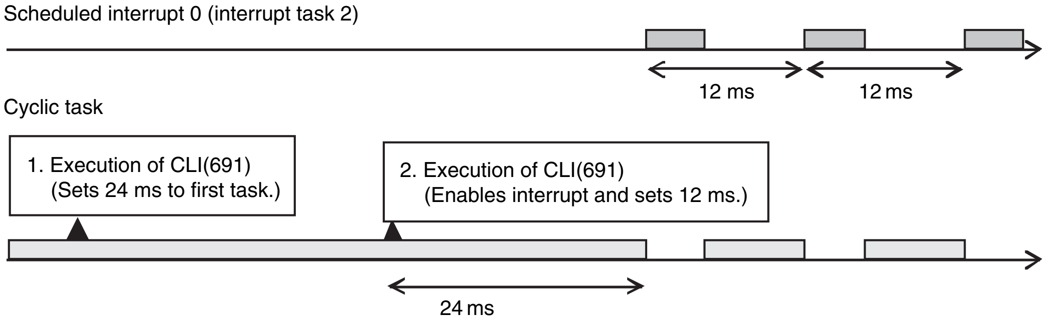
0: Recorded interrupt input retained
1: Recorded interrupt input cleared

● Setting the Time to the First Scheduled Interrupt



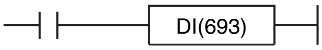
1 When W0.00 goes from OFF to ON, CLI(691) sets the time to the first execution of scheduled interrupt 0 to 24 ms. (In this case, the scheduled interval time unit is set to 1 ms in the PLC Setup.)

2 When W0.01 goes from OFF to ON, CLI(691) sets the time to the first execution of scheduled interrupt 0 to 12 ms, and starts the internal timer. (In this case, the scheduled time interval units are set to 1 ms in the PLC Setup.)



DI

Instruction	Mnemonic	Variations	Function code	Function
DISABLE INTERRUPTS	DI	@DI	693	Disables execution of all interrupt tasks except the power OFF interrupt task for part of a program.

Symbol	DI
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	Not allowed	OK

Flags

Name	Label	Operation
Error Flag	ER	• ON if DI(693) is executed from an interrupt task. • OFF in all other cases.

Related Auxiliary Area Words and Bits

Name	Address	Contents
Disable Setting for Power OFF Interrupts	A530	• A5A5 hex: Enables the Disable Setting for Power OFF Interrupts. Power OFF processing (excluding execution of the Power OFF interrupt task) is masked between the DI(694) and EI(694) instructions, so instructions up to EI(694) are executed. • Other than hex #A5A5: Disable power OFF interrupt processing

Function

DI(693) is executed from the main program to disable all interrupt tasks except the power OFF interrupt (I/O interrupt tasks, scheduled interrupt tasks, external interrupt tasks, input interrupt tasks, and high-speed counter tasks) until EI(694) is executed. DI(693) is used when you do not what to execute interrupt tasks during program execution in cyclic tasks. Interrupt causes that occur while interrupts are disabled are recorded internally, the corresponding interrupt tasks are executed when interrupt tasks are enabled again.

When a CJ2, CS1-H, CJ1-H, CJ1M and CS1D (for Single-CPU System) CPU Unit is being used, power OFF interrupt processing can be disabled simultaneously when A503 (the Disable Setting for Power OFF Interrupts) is set to A5A5 hex. Even if a power interruption is detected after DI(693) has been executed, the CPU Unit will be reset after the program's instructions have been executed in order up to EI(694) or the END(001) instruction in the last task.

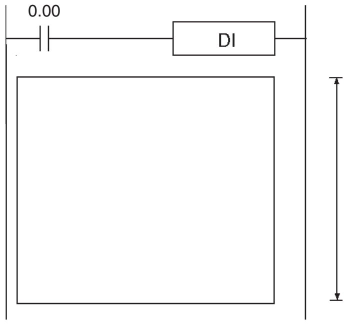
If the power OFF interrupt task is enabled, the CPU Unit will be reset after execution of the power OFF interrupt task. For details, refer to *10-3-2 Instruction Execution for Power Interruptions* in the *SYSMAC CS Series Programmable Controllers Operation Manual*, *10-3-1 Instruction Execution for Power Interruptions* in the *SYSMAC CJ Series Programmable Controllers Operation Manual*, or *A-5-2 Instruction Execution for Power Interruptions* in the *SYSMAC CJ Series CJ2 CPU Unit Software User's Manual*.

Precautions

All interrupt tasks will remain disabled until EI(694) is executed.

DI(693) cannot be executed from an interrupt task.

Example Programming



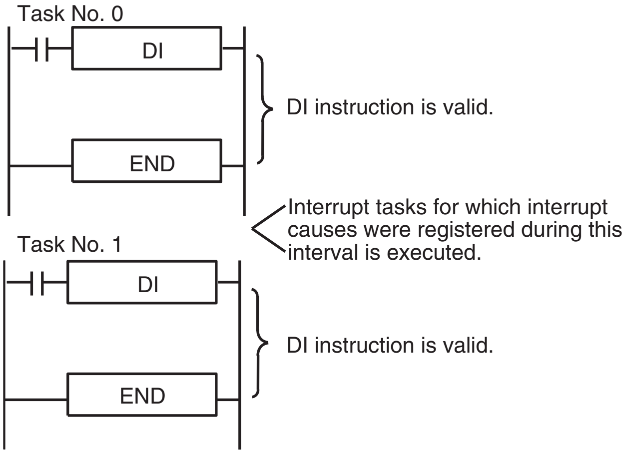
When CIO 0.00 is ON in the following example, DI(693) disables all interrupt tasks other than the power OFF interrupt task.

Note With CS1D CPU Units for Single-CPU Systems or CJ2, CS1-H, CJ1-H, or CJ1M CPU Units:
Power OFF interrupt processing can be disabled at the same time if the power OFF interrupt task is disabled.

● When execution of all interrupts other than power OFF interrupt tasks is prohibited by this instruction

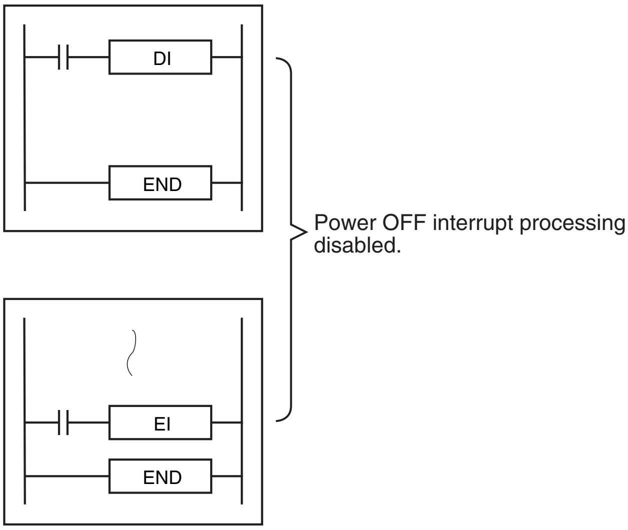
DI(693) cannot be executed for more than one cyclic task. To disable more than one cycle execution task, insert DI(693) in each cyclic task.

However, interrupt tasks are always enabled at the end of a cyclic task, and interrupt tasks for any recorded interrupt causes will be executed at that time.



● When using DI(693) to disable Power OFF Interrupt Processing

It is possible to disable the processing through the cyclic tasks. (The disabled condition is released after the completion of all tasks that were started.)



Note When a CS1D CPU Unit for Single-CPU System or a CJ2, CS1-H, CJ1-H, or CJ1M CPU Unit is being used, the power OFF interrupt task is disabled, and A530 is set to A5A5 hex, the CPU Unit will be reset after execution of EI(694) in the event that a power interruption is detected during execution of the instructions between DI(693) and EI(694).

EI

Instruction	Mnemonic	Variations	Function code	Function
ENABLE INTERRUPTS	EI	---	694	Enables execution of all interrupt tasks that were disabled with DI(693).

	EI
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	Not allowed	OK

Flags

Name	Label	Operation
Error Flag	ER	• ON if EI(694) is executed from an interrupt task. • OFF in all other cases.

Related Auxiliary Area Words and Bits

Name	Address	Contents
Disable Setting for Power OFF Interrupts	A530	A5A5 hex: Enables the Disable Setting for Power OFF Interrupts. Power OFF processing (excluding execution of the Power OFF interrupt task) is masked between the DI(694) and EI(694) instructions, so instructions up to EI(694) are executed. Any other value: Disables the Power OFF Processing mask.

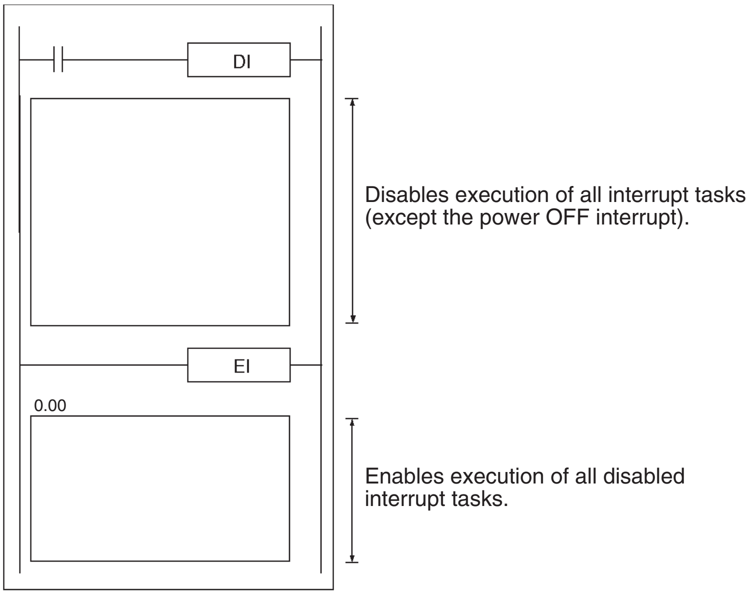
Function

- 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 except the power OFF interrupt (I/O interrupts, scheduled interrupts, and external interrupts).
- When a CJ2, CS1-H, CJ1-H, CJ1M and CS1D CPU Unit for Single-CPU System Unit is being used and power OFF interrupt processing has been disabled with DI(693), EI(694) will also release the hold on power OFF interrupt processing. After DI(593) has been executed, the CPU Unit will not be reset even if a power interruption is detected. The CPU Unit will be reset after all of the instructions between DI(693) and EI(694) have been executed. Refer to *3-21-4 DISABLE INTERRUPTS: DI(693)* for details on using DI(693) to disable power OFF interrupt processing.

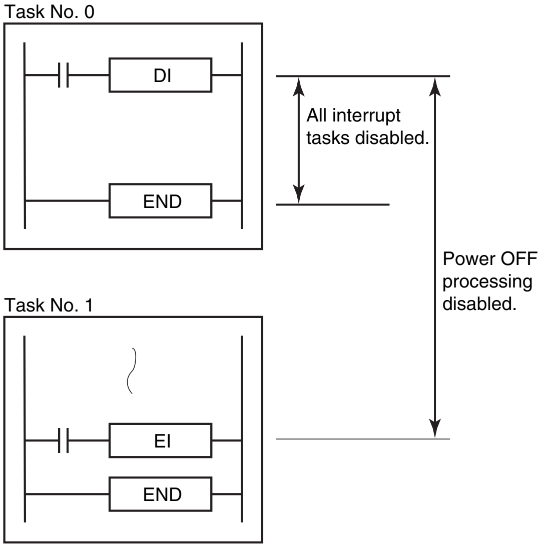
Precautions

- EI(694) does not require an execution condition. It is always executed with an ON execution condition.
- EI(694) enables the interrupt tasks that were disabled by DI(693). It cannot unmask I/O 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.

Example Programming



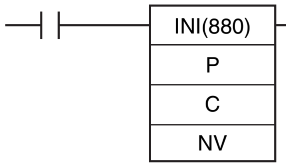
Note When the power OFF interrupt task is disabled for a CJ2, CS1-H, CJ1-H, CJ1M CPU Unit, or CS1D CPU Unit for Single-CPU System, power OFF processing will also be enabled at the same time.



High-speed Counter/Pulse Output Instructions

INI

Instruction	Mnemonic	Variations	Function code	Function
MODE CONTROL	INI	@INI	880	INI(880) is used to start and stop comparison for a comparison table, to change the present value (PV) of a high-speed counter, to change the PV of an input interrupt in counter mode, to change the maximum value of the ring counter (CJ2M only), to change the PV of a pulse output (e.g., to 0 to establish the origin), to stop pulse output, or to change the settings for origin searches/returns (CJ2M only).

	INI
Symbol	 P: Port specifier C: Control data NV: First word with new PV

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
C	Control data	---	1
NV	First word with new PV	DWORD	Variable

P: Port Specifier

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)
0010 hex	High-speed counter 0
0011 hex	High-speed counter 1
0012 hex	High-speed counter 2 (CJ2M only)
0013 hex	High-speed counter 3 (CJ2M only)
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	Input interrupt 4 in counter mode (CJ2M only)
0105 hex	Input interrupt 5 in counter mode (CJ2M only)
0106 hex	Input interrupt 6 in counter mode (CJ2M only)
0107 hex	Input interrupt 7 in counter mode (CJ2M only)
1000 hex	PWM(891) output 0
1001 hex	PWM(891) output 1
1002 hex	PWM output 2 (CJ2M only)
1003 hex	PWM output 3 (CJ2M only)

C: Control Data

C	INI(880) function
0000 hex	Starts comparison.
0001 hex	Stops comparison.
0002 hex	Changes the PV.
0003 hex	Stops pulse output.
0005 hex	Changes origin search/return settings (CJ2M only)
0006 hex	Changes the maximum value of the ring counter (CJ2M only)

NV: First Word with New PV

This operand is used only for changing the PV, changing the maximum ring value, and changing origin search/return settings.

Word	15 to 0
NV	New data

Interrupt Input in Counter Mode: 0000 to FFFF hex

- Changing the PV (C = 0002 hex) or Changing the Maximum Ring Value (C = 0006 hex) (CJ2M Only)

Word	15 to 0
NV	Lower word of new PV
NV+1	Upper word of new PV

For Pulse Output or High-speed Counter Input: 0000 0000 to FFFF FFFF hex

- Changing Origin Search/Return Settings (C = 0005 hex) (CJ2M Only)

Word	15 to 0		Setting range
NV	Origin Search/Return Initial Speed	Rightmost 4 digits	0 to 100,000 pps (100 kpps) (0000 0000 to 0001 86A0 hex)
NV+1		Leftmost 4 digits	
NV+2	Origin Search High Speed	Rightmost 4 digits	1 to 100,000 pps (100 kpps) (0000 0001 to 0001 86A0 hex)
NV+3		Leftmost 4 digits	
NV+4	Origin Search Proximity Speed	Rightmost 4 digits	1 to 100,000 pps (100 kpps) (0000 0001 to 0001 86A0 hex)
NV+5		Leftmost 4 digits	
NV+6	Origin Search Origin Compensation Value	Rightmost 4 digits	–2,147,483,648 to 2,147,483,647 (8000 0000 to 7FFF FFFF hex)
NV+7		Leftmost 4 digits	
NV+8	Origin Search Acceleration Rate		1 to 65,535 pps/4 ms (0001 to FFFF hex)
NV+9	Origin Search Deceleration Rate		1 to 65,535 pps/4 ms (0001 to FFFF hex)
NV+10	Origin Return Target Speed	Rightmost 4 digits	1 to 100,000 pps (100 kpps) (0000 0001 to 0001 86A0 hex)
NV+11		Leftmost 4 digits	
NV+12	Origin Return Acceleration Rate		1 to 65,535 pps/4 ms (0001 to FFFF hex)
NV+13	Origin Return Deceleration Rate		1 to 65,535 pps/4 ms (0001 to FFFF hex)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, C	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
NV	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---			OK				

Flags

Name	Label	Operation
Error Flag	P_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 changing the ring counter maximum value is specified for a high-speed counter that is not set to Ring Mode. • ON if the maximum ring value is changed while comparison is being executed. • ON if a value that exceeds the ring counter maximum value is registered in the comparison table. • 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 exceeds the ring counter maximum value is specified when changing the PV of a high-speed counter in Ring 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 a value that is out of range is specified as the PV for an input interrupt in counter mode. • ON if executed for a port not set for an input interrupt in counter mode. • ON if a new PV is specified for a port that is currently outputting pulses. • ON if parameters are out of range when changing the settings for origin searches and origin returns. • ON if an origin search or origin return setting is charged during an origin search or origin return operation. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

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	Changing the origin search/return settings (0005 hex) (CJ2M only)	Changing the maximum ring value (0006 hex) (CJ2M only)
0000 or 0001 hex: Pulse output	Not allowed.	Not allowed.	OK	OK	OK	Not allowed.
0010 or 0013 hex: High-speed counter input	OK	OK	OK	Not allowed.	Not allowed.	OK
0100 to 0107 hex: Input interrupt in counter mode	Not allowed.	Not allowed.	OK	Not allowed.	Not allowed.	Not allowed.
1000 or 1001 hex: PWM (891) output	Not allowed.	Not allowed.	Not allowed.	OK	Not allowed.	Not allowed.

● 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)

Port and mode			Operation	Setting range
Pulse output (P = 0000 0003 hex)			The present value of the pulse output is changed. The new value is specified in NV. **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. **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.	0000 to FFFF hex (0 to 65,535) **Note:** An error will occur if a value outside this range is specified.

● Stopping Pulse Output (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.

● Changing Origin Search/Return Settings (C = 0005 Hex) (CJ2M Only)

If C is 0007 hex, INI(880) changes the settings for origin searches/returns during operation. The initial settings for origin searches/returns are set in the PLC Setup. INI(880) is used to temporarily change a speed or acceleration/deceleration rate in an application.

Note The settings in the PLC Setup are not changed. When power is turned ON, the settings in the PLC Setup will be used.

The following parameters can be changed with INI(880).

All of these parameters can be changed at the same time. All parameters must be specified even when changing only one of them.

If invalid parameters are specified, an instruction error will occur and the parameters will not be changed. If an origin search or origin return is executed after such an error occurs, the previous parameters will be used in the operation.

Origin search/return parameters that can be changed with INI(880)		NV (first word with new value)
Basic setting	Origin Search/Return Initial Speed	NV+1, NV
Origin search settings	High Speed	NV+3, NV+2
	Proximity Speed	NV+5, NV+4
	Origin Compensation	NV+7, NV+6
	Acceleration Rate	NV+8
	Deceleration Rate	NV+9
Origin return settings	Target Speed	NV+11, NV+10
	Acceleration Rate	NV+12
	Deceleration Rate	NV+13

● Changing the Maximum Ring Value (C = 0006 Hex) (CJ2M Only)

If C is 0006 hex, INI(880) changes the maximum ring value of the high-speed counter during operation. The initial value for the maximum ring value is set in the PLC Setup. INI(880) is used to temporarily change the maximum ring value in an application.

The following restrictions apply.

- The maximum ring value can be changed only when the comparison operation is stopped.
- CTBL(882) must be executed again to restart the comparison operation after the maximum ring value is changed.

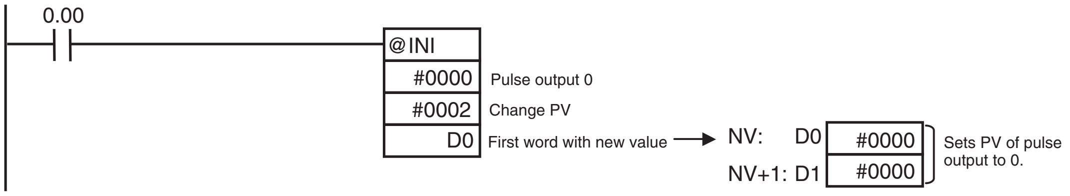
Note The settings in the PLC Setup are not changed. When power is turned ON, the settings in the PLC Setup will be used.

When the maximum ring value is changed, the PV of the high-speed counter will be cleared to zero.

Example Programming

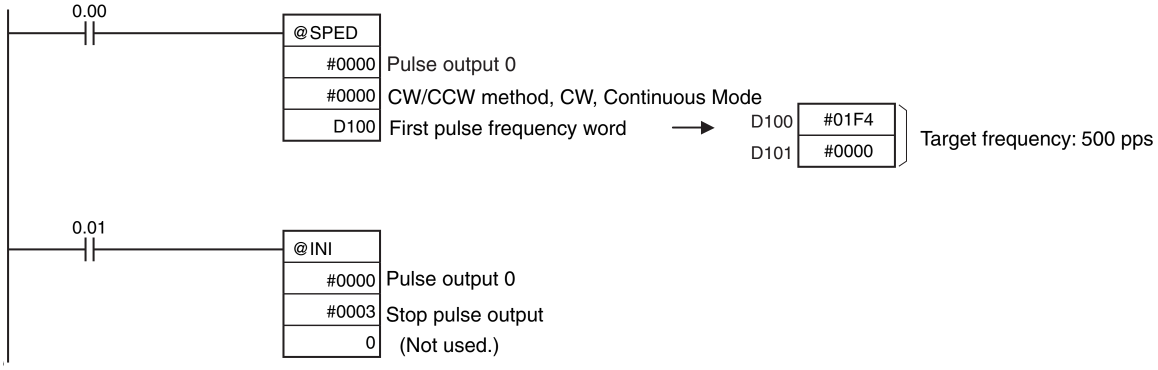
Example: Changing the PV for Pulse Output (C = 0002 Hex)

When CIO 0.00 turns ON in the following example, INI(880) changes the PV of pulse output 0 to 0.



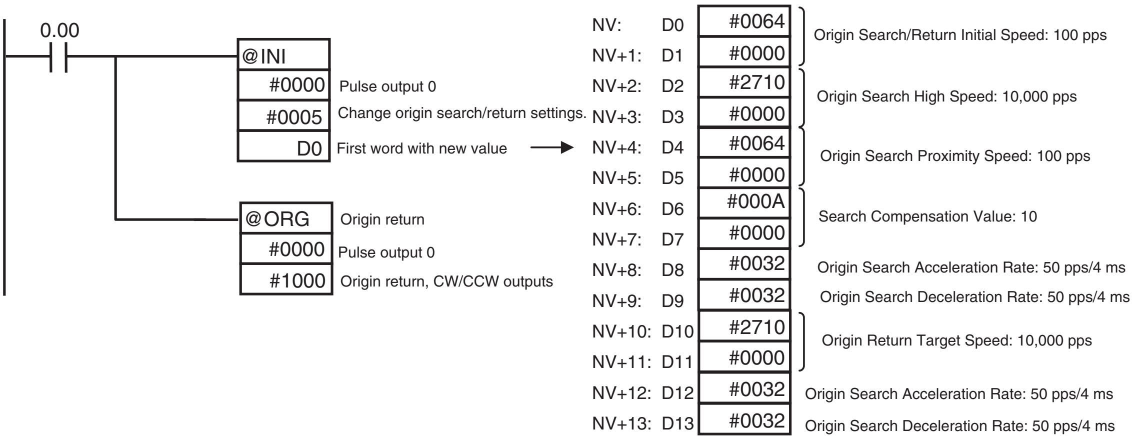
Example: Stopping Pulse Output (C = 0003 Hex)

When CIO 0.00 turns ON in the following example, SPED(885) starts outputting pulses from pulse output 0 in Continuous Mode at 500 pps. When CIO 0.01 turns ON, pulse output is stopped by INI(880).



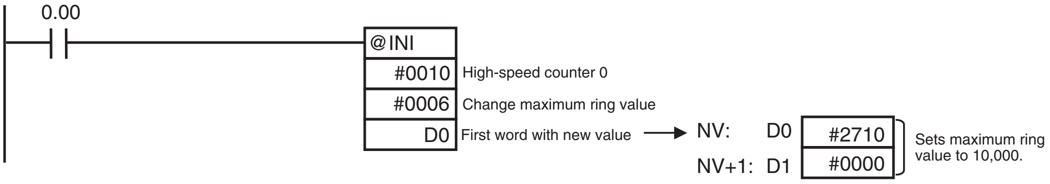
Example: Changing Origin Search/Return Settings (C = 0005 Hex) (CJ2M Only)

When CIO 0.00 turns ON in the following example, INI(880) changes the origin search/return settings. Then an origin return is executed with ORG(889).



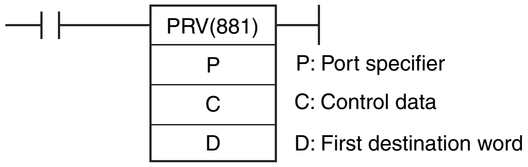
Example: Changing the Maximum Ring Value (C = 0006 Hex) (CJ2M Only)

When CIO 0.00 turns ON in the following example, INI(880) changes the maximum ring value.



PRV

Instruction	Mnemonic	Variations	Function code	Function
HIGH-SPEED COUNTER PV READ	PRV	@PRV	881	PRV(881) reads the High-speed counter PV and pulse output PV and interrupt input PV in counter mode.

	PRV
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
C	Control data	---	1
D	First destination word	WORD	Variable

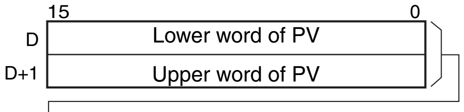
P: Port Specifier

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)
0010 hex	High-speed counter 0
0011 hex	High-speed counter 1
0012 hex	High-speed counter 2 (CJ2M only)
0013 hex	High-speed counter 3 (CJ2M only)
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	Input interrupt 4 in counter mode (CJ2M only)
0105 hex	Input interrupt 5 in counter mode (CJ2M only)
0106 hex	Input interrupt 6 in counter mode (CJ2M only)
0107 hex	Input interrupt 7 in counter mode (CJ2M only)
1000 hex	PWM(891) output 0
1001 hex	PWM(891) output 1
1002 hex	PWM output 2 (CJ2M only)
1003 hex	PWM output 3 (CJ2M only)

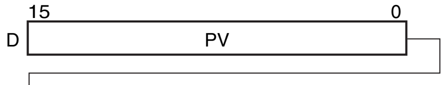
C: Control Data

C	PRV(881) function
0000 hex	Reads the PV.
0001 hex	Reads status.
0002 hex	Reads range comparison results.
00□3 hex	P = 0000 to 0003: Reads the output frequency of pulse output 0 to 3. C = 0003 hex: Standard operation P = 0010: Reads the frequency of high-speed counter input 0 (high-speed counter 0 only). C = 0003 hex: Standard operation C = 0013 hex: 10-ms sampling method for high frequency (supported only by CJ1M CPU Units with unit version 3.0 or later or CJ2M CPU Units) C = 0023 hex: 100-ms sampling method for high frequency (supported only by CJ1M CPU Units with unit version 3.0 or later or CJ2M CPU Units) C = 0033 hex: 1-s sampling method for high frequency (supported only by CJ1M CPU Units with unit version 3.0 or later or CJ2M CPU Units)

D: First Destination Word



2-word PV
Pulse output PV, high-speed counter input PV, pulse output frequency, high-speed counter input frequency, high-speed counter input for high-speed counter input 0, or range comparison results for 1 to 32 ranges



1-word PV
Interrupt input PV in counter mode, status, range comparison results for 8 ranges

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, C	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---			OK				

Flags

Name	Label	Operation
Error Flag	P_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 input interrupt in counter mode. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.
Carry Flag (CJ2M CPU Units Only)	P_CY	• ON if a high-frequency method is specified for reading the frequency of high-speed counter 0 and the data could not be read dependably within the sampling time because the counter value was reset. • OFF in all other cases.

Function

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	00□3 hex: Read frequency
(0000 to 0003 hex): Pulse output	OK	OK	Not allowed.	Refer to the following table.
(0010 to 0013 hex): High-speed counter input	OK	OK	OK	
(0100 to 0107 hex): Interrupt input in counter mode	OK	Not allowed.	Not allowed.	
(1000 to 1003 hex): PWM (891) output	Not allowed.	OK	Not allowed.	

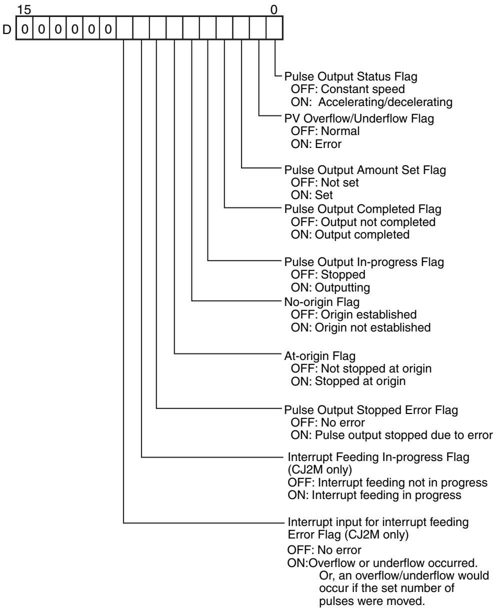
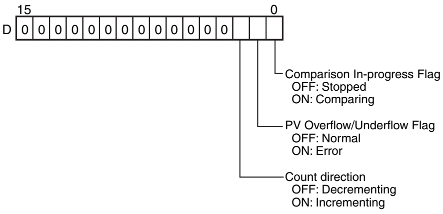
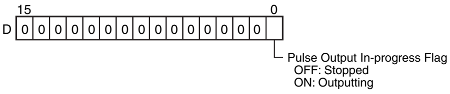
P: Port specifier	00□3 hex: Read frequency			
	0003 hex: Pulse output read high-speed counter frequency	0013 hex: 10-ms sampling method	0013 hex: 100-ms sampling method	0013 hex: 1-s sampling method
(0000 to 0003 hex): Pulse output	OK (See note.)	Not allowed.	Not allowed.	Not allowed.
(0010 to 0013 hex): High-speed counter input	OK (high-speed counter 0 only)	OK (See note.) (high-speed counter 0 only)	OK (See note.) (high-speed counter 0 only)	OK (See note.) (high-speed counter 0 only)
(0100 to 0107 hex): Interrupt input in counter mode	Not allowed.	Not allowed.	Not allowed.	Not allowed.
(1000 to 1003 hex): PWM (891) output	Not allowed.	Not allowed.	Not allowed.	Not allowed.

Note This function is supported only by CJ1M CPU Units with unit version 3.0 or later and CJ2M CPU Units.

● Reading a PV (C = 0000 hex)

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)
Input interrupt in counter mode (P = 0100 to 0107 hex)		The present value of the input interrupt is stored in D.	0000 to FFFF hex (0 to 65,535)

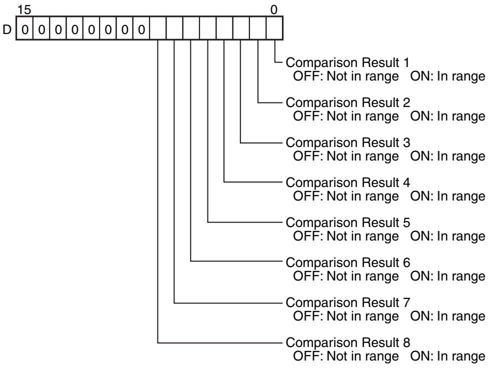
Reading Status (C = 0001 hex)

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(891) output	The PWM(891) output is stored in D.	

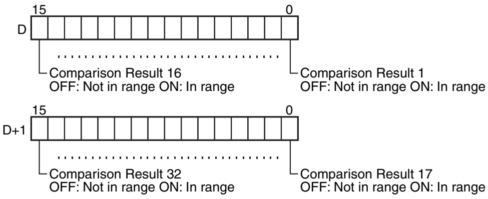
Reading the Results of Range Comparison (C = 0002 hex)

If C is 0002 hex, PRV(881) reads the results of range comparison and stores it in D as shown in the following diagram.

Results for Eight Ranges



Results for 1 to 32 Ranges (CJ2M Only)



Note: The comparison result will be OFF for any of the 32 counter input ranges that are not specified.

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.

0000 or 0001 hex (Reading the frequency of pulse output 0 to 3)

0000 0000 to 0001 86A0 hex (0 to 100,000)

0010 hex (Reading the frequency of high-speed counter 0)
Counter input method: Any input method other than 4× differential phase mode:

Result = 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.

Counter input method: 4× differential phase mode:

Result = 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 200,000 decimal.

Pulse Frequency Calculation Methods

- Pulse Output:
  The output frequency can be read by executing the PRV(881) instruction.
- High-speed Counter:

  When the CPU Unit is a CJ1M CPU Unit with unit version 3.0 or later or a CJ2M CPU Unit, there are two ways to calculate the frequency of pulses input to high-speed counter 0.

  - 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).

- High-frequency Calculation Method: In this case, the counting method is switched at high and low frequencies.
  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 10% max. at 1 kHz.
100 ms	0023 hex	Counts the number of pulses every 100 ms. The error is 1% max. at 1 kHz.
1 s	0033 hex	Counts the number of pulses every 1 s. The error is 0.1% max. at 1 kHz.

At frequencies below 1 kHz, the Standard Calculation Method is used, regardless of the sampling time setting.

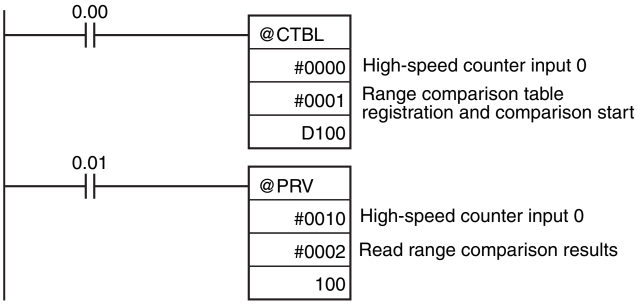
The update period for the frequency is the same as the sampling time. Following characteristics for changes in the frequency will deteriorate as the sampling time increases.

Precautions

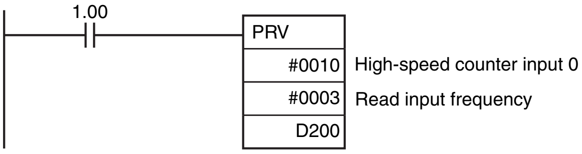
If the counter is reset when P is 0010 hex (high-speed counter 0) and C is 0013, 0023, or 0033 hex (sampling method for high frequency), the data read during the sampling time when the counter was reset will not be dependable.
When this is performed for a CJ2M CPU Unit, the Carry Flag (P_CY) will turn ON when the PRV(881) instruction is executed.

Example Programming

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 0100.

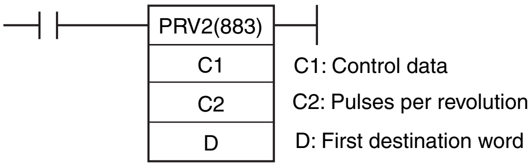


When CIO 1.00 turns ON in the following programming example, PRV(881) reads the frequency of the pulse being input to high-speed counter 0 at that time and stores it as a hexadecimal value in D200 and D201.



PRV2

Instruction	Mnemonic	Variations	Function code	Function
COUNTER FREQUENCY CONVERT	PRV2	@PRV2	883	Reads the pulse frequency input from a high-speed counter and either converts the frequency to a rotational speed (number of revolutions) or converts the counter PV to the total number of revolutions. The result is output to the destination words as 8-digit hexadecimal. The frequency measurement function can be used with high-speed counter 0 only.

	PRV2
Symbol	

Applicable Program Areas

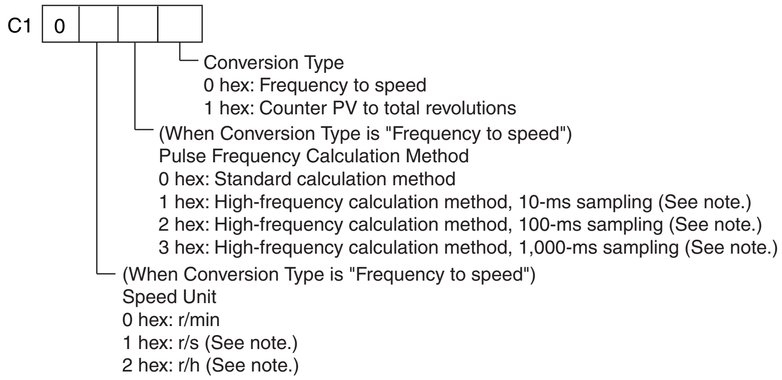
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C1	Control data	---	1
C2	Pulses per revolution	UINT	1
D	First destination word	UDINT	2

C1: Control data

C1	PRV2(883) function
0□*0 hex	Converts frequency to rotation speed. (The second digit of C (□) specifies the units and the third digit (*) specifies the frequency calculation method.)
0001 hex	Converts counter PV to total number of revolutions.



Conversion Type
0 hex: Frequency to speed
1 hex: Counter PV to total revolutions

(When Conversion Type is "Frequency to speed")
Pulse Frequency Calculation Method
0 hex: Standard calculation method
1 hex: High-frequency calculation method, 10-ms sampling (See note.)
2 hex: High-frequency calculation method, 100-ms sampling (See note.)
3 hex: High-frequency calculation method, 1,000-ms sampling (See note.)

(When Conversion Type is "Frequency to speed")
Speed Unit
0 hex: r/min
1 hex: r/s (See note.)
2 hex: r/h (See note.)

Note Supported by CJM1 CPU Unit Ver. 3.0 or later only.

C2: Pulses per Revolution

Specifies the number of pulses per revolution (0001 to FFFF hex).

D: First Destination Word

	15 0
D	Conversion result (Rightmost 4 digits)
D+1	Conversion result (Leftmost 4 digits)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C1	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
C2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK		OK				
D											---	---						

Flags

Name	Label	Operation
Error Flag	P_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. • ON when functions are specified for inputs on a Pulse I/O Module when a Pulse I/O Module is not mounted to the CJ2M CPU Unit. • OFF in all other cases.
Carry Flag (CJ2M CPU Units Only)	P_CY	• ON if a high-frequency method is specified for reading the frequency of high-speed counter 0 and the data could not be read dependably within the sampling time because the counter value was reset. • OFF in all other cases.

Function

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 (CJM1 CPU Unit Ver. 3.0 or later only)
  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/h (CJM1 CPU Unit Ver. 3.0 or later only)
  When the second digit of C (□) is 2, PRV2(883) calculates the rotation speed in r/h from the frequency data and pulses/revolution setting.
  Rotation speed (r/h) = (Frequency ÷ Pulses/revolution) × 60 × 60
- Range of Conversion Results
  - Counter input method: Any method besides 4× differential phase mode
    Frequency = 00000000 to 000186A0 hex (0 to 100,000)

Note If a frequency that exceeds 100 kHz is input, the maximum output value (100,000 decimal) will be held.

  - Counter input method: 4× differential phase mode
    Frequency = 00000000 to 00030D40 hex (0 to 200,000)

Note If a frequency that exceeds 200 kHz is input, the maximum output value (200,000 decimal) will be held.

(2) Frequency Calculation Method

When the CPU Unit is a CJ1M CPU Unit with unit version 3.0 or later or a CJ2M CPU Unit, there are two ways to calculate the frequency of pulses input to high-speed counter 0.

- 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).
- High-frequency Calculation Method
  In this case, the counting method is switched at high and low frequencies. (Supported by CJ1M CPU Units with unit version 3.0 or later or CJ2M CPU Units only)
  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.

At frequencies below 1 kHz, the Standard Calculation Method is used, regardless of the sampling time setting.

The update period for the frequency is the same as the sampling time. Following characteristics for changes in the frequency will deteriorate as the sampling time increases.

● 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.

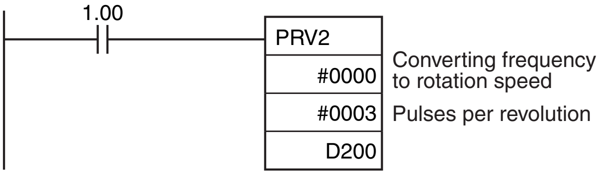
Conversion result = Counter PV ÷ Pulses/revolution

Precautions

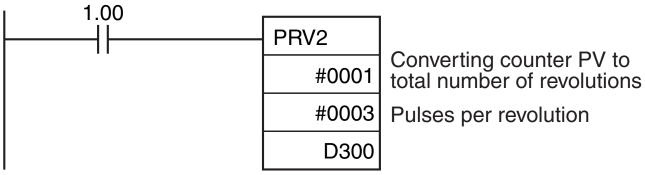
If the counter is reset when C1 specifies frequency-rotational speed conversion for a high frequency, the data read during the sampling time when the counter was reset will not be dependable.
When this is performed for a CJ2M CPU Unit, the Carry Flag (P_CY) will turn ON when the PRV2(883) instruction is executed.

Example Programming

When CIO 1.00 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.

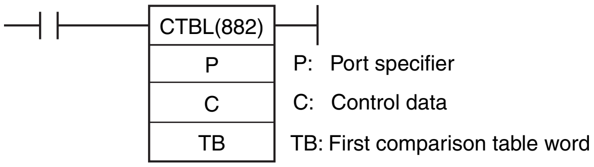


When CIO 1.00 is ON in the following programming example, PRV2(883) reads the counter PV, converts that value to number of revolutions, and outputs the hexadecimal result to D301 and D300.



CTBL

Instruction	Mnemonic	Variations	Function code	Function
REGISTER COMPARISON TABLE	CTBL	@CTBL	882	CTBL(882) is used to register a comparison table and perform comparisons for a high-speed counter PV.

	CTBL
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
C	Control data	---	1
TB	First comparison table word	LWORD	Variable

P: Port specifier

P	Port
0000 hex	High-speed counter 0
0001 hex	High-speed counter 1
0002 hex	High-speed counter 2 (CJ2M only)
0003 hex	High-speed counter 3 (CJ2M only)

C: Control data

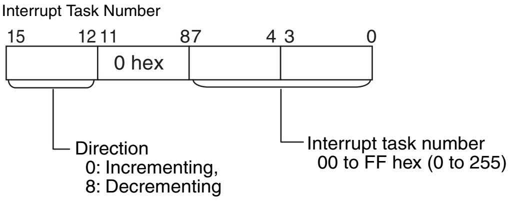
C	CTBL(882) function
0000 hex	Registers a target value comparison table and starts comparison.
0001 hex	Registers a range comparison table with 8 ranges and starts comparison.
0002 hex	Registers a target value comparison table. Comparison is started with INI(880).
0003 hex	Registers a range comparison table with 8 ranges, but does not perform comparison.
0004 hex	Registers a range comparison table with 1 to 32 ranges, and starts comparison (CJ2M only).
0005 hex	Registers a range comparison table with 1 to 32 ranges, but does not perform comparison (CJ2M only).

TB: First comparison table word

● Specifying a Target Value Comparison Table (C = 0000 or 0002 Hex)

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.

	15 … 0	
TB	Number of target values	0001 to 30 hex (1 to 48 target values)
TB+1	Lower word of target value 1	00000000 to FFFFFFFF hex
TB+2	Upper word of target value 1	
TB+3	Interrupt task number for target value 1	
⋮	⋮	
TB+142	Lower word of target value 48	00000000 to FFFFFFFF hex
TB+143	Upper word of target value 48	
TB+144	Interrupt task number for target value 48	

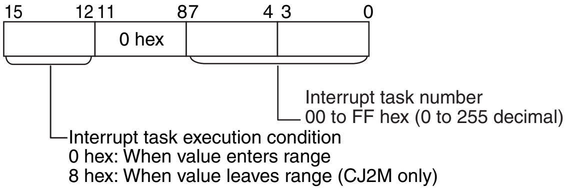


● Specifying a Range Comparison Table with 8 Ranges (C = 0001 or 0003 Hex)

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. The ranges are set using upper and lower limits.

	15 … 0	
TB	Lower word of range 1 lower limit	0000 0000 to FFFF FFFF hex (See note.)
TB+1	Upper word of range 1 lower limit	
TB+2	Lower word of range 1 upper limit	0000 0000 to FFFF FFFF hex (See note.)
TB+3	Upper word of range 1 upper limit	
	Range 1 interrupt task number	
⋮	⋮	
TB+35	Lower word of range 8 lower limit	0000 0000 to FFFF FFFF hex (See note.)
TB+36	Upper word of range 8 lower limit	
TB+37	Lower word of range 8 upper limit	0000 0000 to FFFF FFFF hex (See note.)
TB+38	Upper word of range 8 upper limit	
TB+39	Range 8 interrupt task number	

Interrupt task number



AAAA hex: Do not execute interrupt task. (Flag output only)
FFFF hex: Ignore the settings for this range.

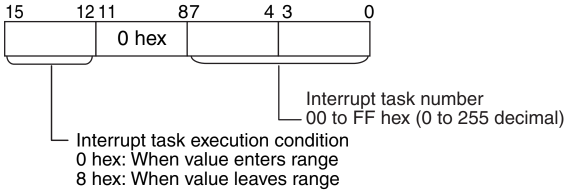
Note Always set the upper limit greater than or equal to the lower limit for any one range.

● Specifying a Range Comparison Table with 1 to 32 Ranges (C = 0004 or 0005 Hex) (CJ2M Only)

The length of the comparison table is determined by the number of target values specified in TB. The table can be between 6 and 161 words long. Set valid values for the entire table specified by TB. (It is not possible to skip settings for any ranges.)

Word	Contents	Setting range
TB	Number of ranges	0001 to 0020 hex (1 to 32 decimal)
TB+1	Lower word of range 1 lower limit	0000 0000 to FFFF FFFF hex
TB+2	Upper word of range 1 lower limit	
TB+3	Lower word of range 1 upper limit	0000 0000 to FFFF FFFF hex
TB+4	Upper word of range 1 upper limit	
TB+5	Range 1 interrupt task number	
	•	
	•	
	•	
TB+156	Lower word of range 32 lower limit	0000 0000 to FFFF FFFF hex
TB+157	Upper word of range 32 lower limit	
TB+158	Lower word of range 32 upper limit	0000 0000 to FFFF FFFF hex
TB+159	Upper word of range 32 upper limit	
TB+160	Range 32 interrupt task number	

Interrupt task number



AAAA hex: Do not execute interrupt task. (Flag output only)
FFFF hex: Ignore the settings for this range.

Note Always set the upper limit greater than or equal to the lower limit for any one range.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, C	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
TB	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---			OK				

Flags

Name	Label	Operation
Error Flag	P_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 number of ranges specified for range comparison (1 to 32) is set to 0. • ON if the number of ranges specified for range comparison (1 to 32) is set to a value higher than 32. • 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. • ON if CTBL(882) is executed from an interrupt task for the same port when an interrupt occurs during CTBL(882) execution. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

CTBL(882) registers a comparison table and starts target-value or range comparison for the high-speed counter specified in P and the method specified in C.

● Registering a Comparison Table (C = 0002, 0003, 0005 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, 0001, 0004 Hex)

If C is set to 0000 or 0001 hex, a comparison table will be registered, and comparison will be started.

- 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. 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.
- 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.
  If bit 00 is ON, the PV will be compared to the target value 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 PV of the high-speed counter is compared to the specified ranges and the result is output to flags. Also, a specified interrupt can be executed when the PV is outside a range (CJ2M only) or in a range.

- The range comparison table contains 1 to 32 ranges (CJ2M only) or 8 ranges. Overlapping ranges can be specified. An interrupt task number is set for each range.
- The same interrupt task number can be specified for more than one target value.
- The interrupt task is executed either when the PV enters the range or leaves the range. (This is specified with the MSB when specifying the interrupt task number. The interrupt task is executed only once when the PV enters or leaves 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.

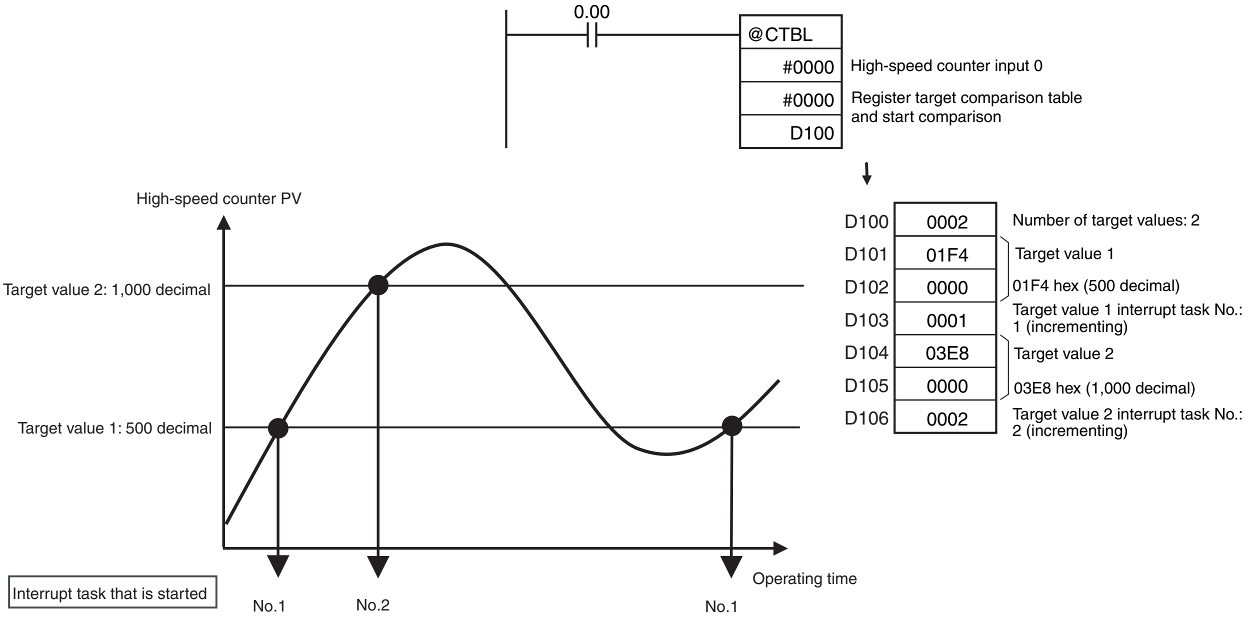
Note An error will occur if the upper limit is less than the lower limit for any one range.

- 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 in the Auxiliary Area.

- For range comparisons, the interrupt task for any one range will be executed only once each time the comparison value enters or leaves the range. It will not be executed again until the condition for execution is no longer met and then met again. The Range Comparison Condition Met Flag, however, will be ON whenever the comparison value is within the range regardless of the out of range/in range specification.
- Set the interrupt task number to FFFF hex to disable the range. Any range set to FFFF hex will be ignored.

Example Programming

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.

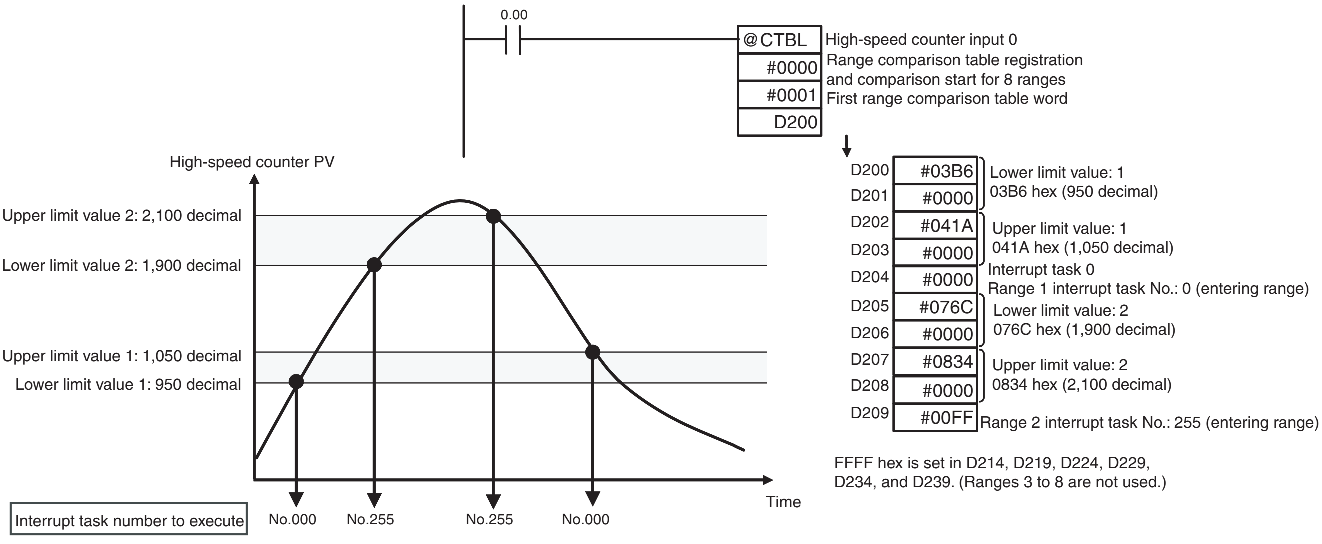


D100	0002	Number of target values: 2
D101	01F4	Target value 1 01F4 hex (500 decimal)
D102	0000	
D103	0001	Target value 1 interrupt task No.: 1 (incrementing)
D104	03E8	Target value 2 03E8 hex (1,000 decimal)
D105	0000	
D106	0002	Target value 2 interrupt task No.: 2 (incrementing)

● Example for Range Comparison with Eight Ranges

When CIO 0.00 turns ON in the following programming example, CTBL(882) registers a range comparison table using upper and lower limits for high-speed counter 0 and starts comparison.

The corresponding interrupt task is called and executed when the PV is within the limits set for range 1 or 2.

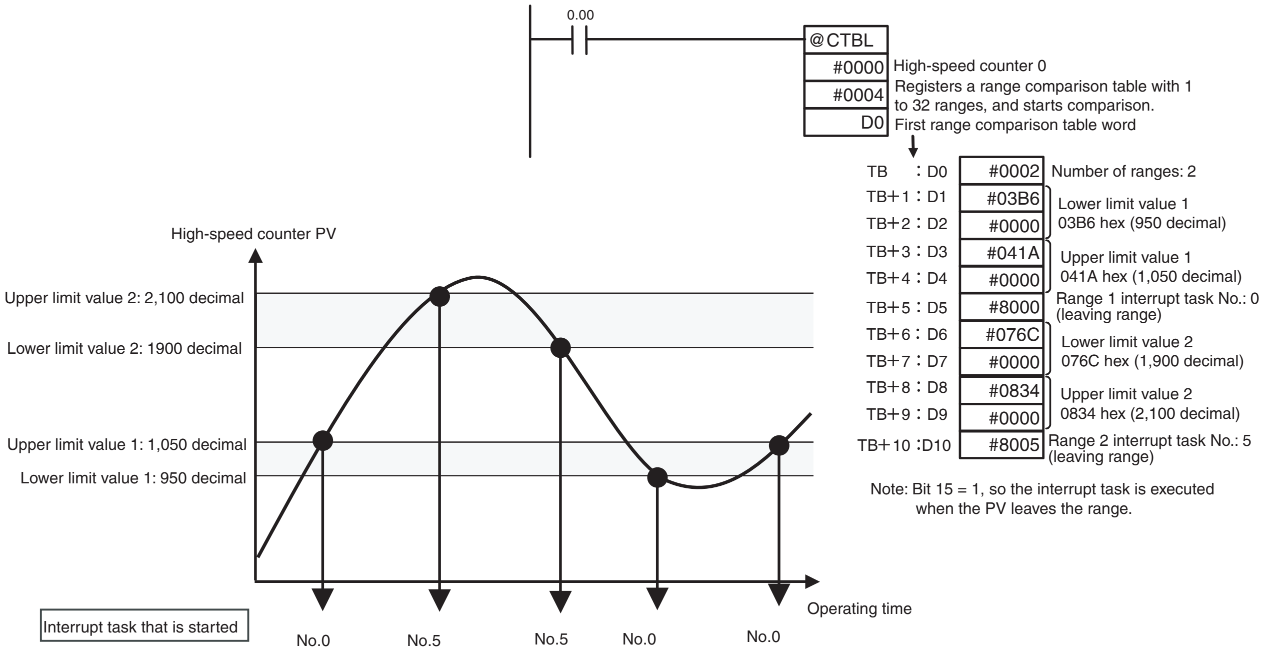


D200	#03B6	Lower limit value: 1 03B6 hex (950 decimal)
D201	#0000	
D202	#041A	Upper limit value: 1 041A hex (1,050 decimal)
D203	#0000	
D204	#0000	Interrupt task 0 Range 1 interrupt task No.: 0 (entering range)
D205	#076C	Lower limit value: 2 076C hex (1,900 decimal)
D206	#0000	
D207	#0834	Upper limit value: 2 0834 hex (2,100 decimal)
D208	#0000	
D209	#00FF	Range 2 interrupt task No.: 255 (entering range)

FFFF hex is set in D214, D219, D224, D229, D234, and D239. (Ranges 3 to 8 are not used.)

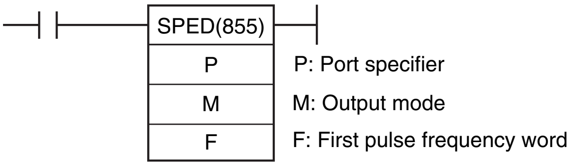
Example for Range Comparison for 1 to 32 Ranges (CJ2M Only)

When CIO 0.00 turns ON in the following programming example, CTBL(882) registers a range comparison table using upper and lower limits for high-speed counter 0 and starts comparison. When the PV of the high-speed counter leaves the upper/lower limits for range 1, interrupt task 0 is executed. When the PV leaves the upper/lower limits for range 2, interrupt task 5 is executed.



SPED

Instruction	Mnemonic	Variations	Function code	Function
SPEED OUTPUT	SPED	@SPED	885	SPED(885) is used to set the output pulse frequency for a specific port and start pulse output without acceleration or deceleration.

	SPED
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

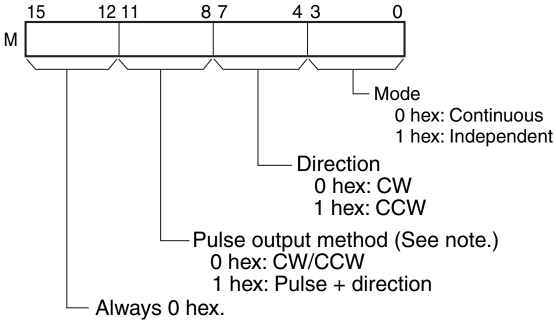
Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
M	Output mode	---	1
F	First pulse frequency word	UDINT	2

P: Port specifier

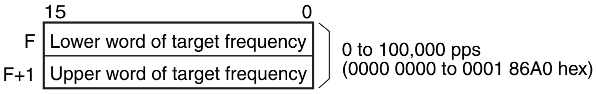
P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)

M: Output mode



Note Use the same pulse output method when using pulse outputs 0 and 1 at the same time and when using 2 and 3 at the same time.

F: First pulse frequency word



The value of F and F+1 sets the pulse frequency in pps.

● Operand Specifications

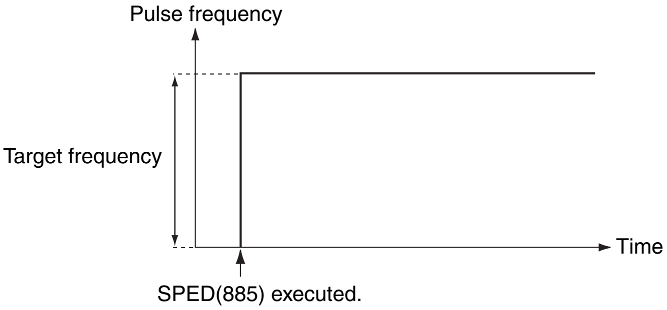
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, M	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
F	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK				OK				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the specified range for P, M, or F is exceeded. • ON if ACC(888), PLS2(887), ORG(889) or IFEED(892) is already being executed to control pulse output for the specified port. • ON if SPED(885) 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. • ON if SPED(885) is executed in independent mode and the number of pulses is not set. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

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.



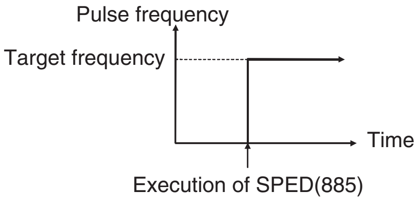
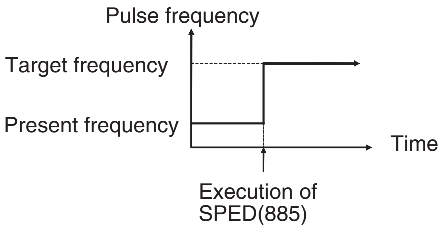
In continuous mode, pulse output will continue until stopped from the program. In independent mode, pulse output will stop automatically when the number of pulses set with PULS(886) in advance have been output.

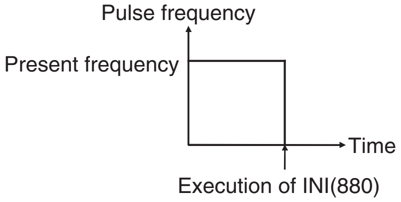
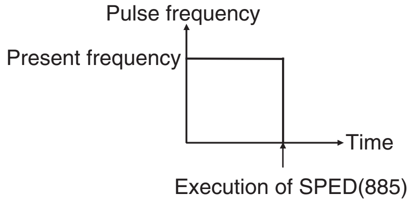
An error will occur if the mode is changed between independent and continuous mode while pulses are being output.

● 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.

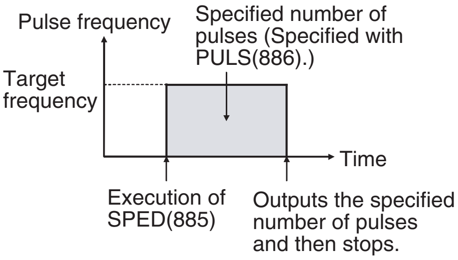
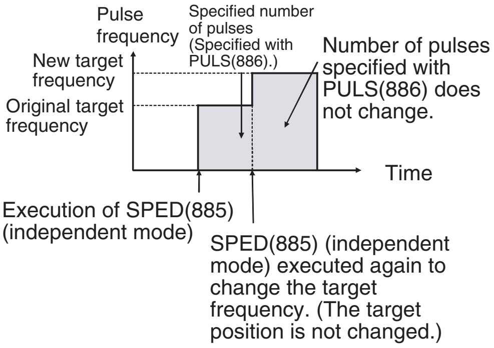
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified speed	Changing the speed (frequency) in one step		Outputs pulses at a specified frequency.	SPED(885) (Continuous)
Changing settings	To change speed in one step	Changing the speed during operation		Changes the frequency (higher or lower) of the pulse output in one step.	SPED(885) (Continuous) ↓ SPED(885) (Continuous)

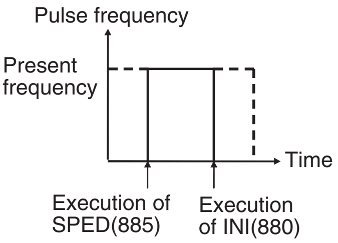
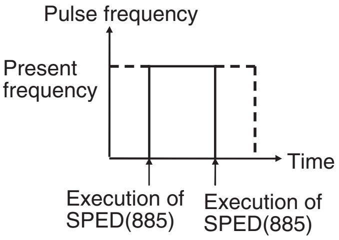
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output	Stop pulse output	Immediate stop		Stops the pulse output immediately.	SPED(885) (Continuous) ↓ INI(880)
	Stop pulse output	Immediate stop		Stops the pulse output immediately.	SPED(885) (Continuous) ↓ SPED(885), ACC(888) (Continuous, Target frequency of 0 pps)

● Independent Mode Positioning

When independent mode operation is started, pulse output will be continued until the specified number of pulses has been output.

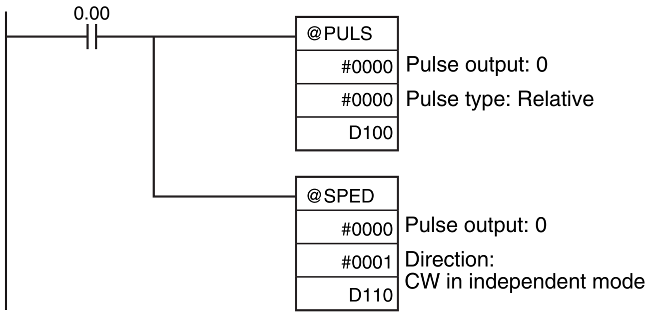
Note
- Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.
- The number of output pulses must be set each time output is restarted.
- The number of output pulses must be set in advance with PULS(881). Pulses will not be output for SPED(885) if PULS(881) is not executed first.
- 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.

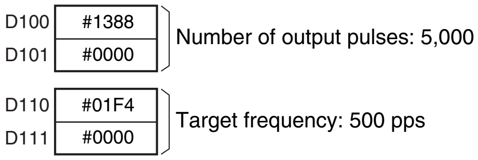
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified speed	Positioning without acceleration or deceleration		Starts outputting pulses at the specified frequency and stops immediately when the specified number of pulses has been output. **Note** The target position (number of pulses) cannot be changed during positioning (pulse output).	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)

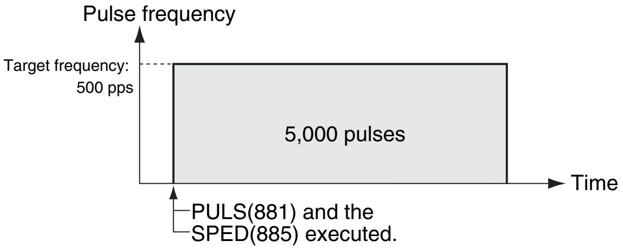
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output	To stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses setting.	PULS(886) ↓ SPED(885) (Independent) ↓ INI(880)
	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), ACC(888) (Independent, Target frequency of 0 pps)

Example Programming

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 pps.

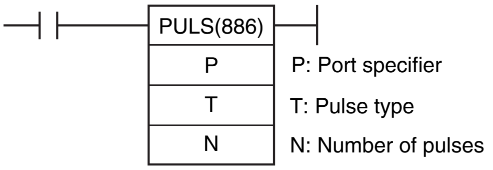






PULS

Instruction	Mnemonic	Variations	Function code	Function
SET PULSES	PULS	@PULS	886	PULS(886) is used to set the pulse output amount (number of output pulses).

	PULS
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
T	Pulse type	---	1
N	Number of pulses	DINT	2

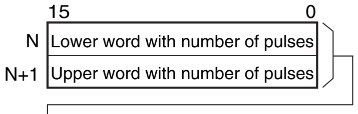
P: Port specifier

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)

T: Pulse type

T	Pulse type
0000 hex	Relative
0001 hex	Absolute

N and N+1: Number of pulses



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.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, T	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK				OK				

Flags

Name	Label	Operation
Error Flag	P_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. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

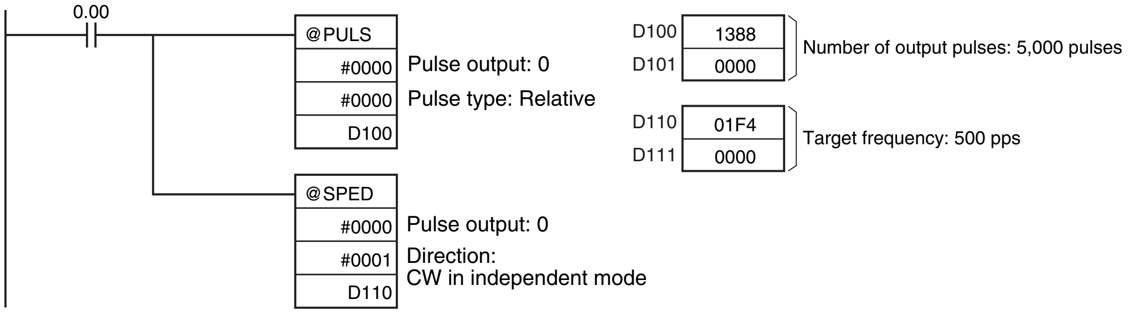
PULS(886) sets the pulse type and number of pulses specified in T and N for the port specified in P. Actual output of the pulses is started later in the program using SPED(885) or ACC(888) in independent mode.

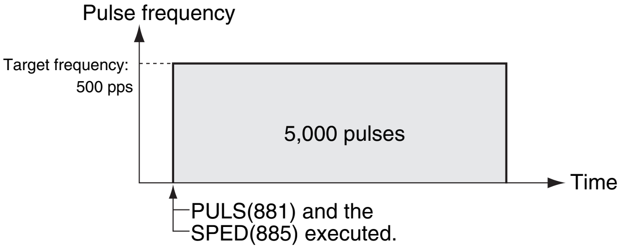
Note

- 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.

Example Programming

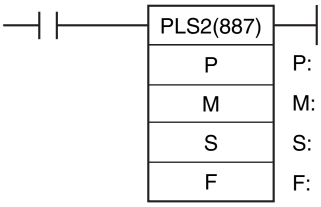
When CIO 0.00 turns ON in the following programming example, PULS(886) sets the number of output pulses for pulse output 0. An absolute value of 5,000 pulses is set. SPED(885) is executed next to start pulse output using the CW/CCW method in the clockwise direction in independent mode at a target frequency of 500 pps.





PLS2

Instruction	Mnemonic	Variations	Function code	Function
PULSE OUTPUT	PLS2	@PLS2	887	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.

	PLS2
Symbol	 P: Port specifier M: Output mode S: First word of settings table F: First word of starting frequency

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

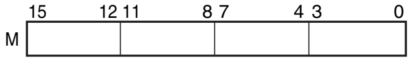
Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
M	Output mode	---	1
S	First word of settings table	WORD	6
F	First word of starting frequency	UDINT	2

P: Port Specifier

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)

M: Output Mode



Relative/absolute specifier
0 hex: Relative pulses
1 hex: Absolute pulses

Direction
0 hex: CW
1 hex: CCW

Pulse output method (See note 1.)
0 hex: CW/CCW
1 hex: Pulse + direction

Stopping operation for reversal specification (See note 2.)/operation compensation for parameters changes (See note 3.)
0 hex: Deceleration stop when reversing and no operation compensation
4 hex: Deceleration stop when reversing and operation compensation
8 hex: Immediate stop when reversing and no operation compensation
C hex: Immediate stop when reversing and operation compensation

Note: With the CJ1M-CPU2□, only 0 hex can be specified.

Note 1 Use the same pulse output method when using both pulse outputs 0 and 1.

2 The stopping method to use when reversing operation can be specified.

3 If a constant speed cannot be achieved after parameters are changed, operation compensation can be set to continue operation.

S: First Word of Settings Table

	15 0
S1	Acceleration rate
S1+1	Deceleration rate

CJ1M CPU Units, no unit version
1 to 2000 pps (#0001 to 07D0)
CJ1M CPU Unit with unit version 2.0 or later or CJ2M CPU Unit
1 to 65535 pps (#0001 to FFFF)

Specify the increase or decrease in the frequency per pulse control period (4 ms).

S1+2	Lower word with target frequency
S1+3	Upper word with target frequency

1 to 100,000 pps
(0000 0000 to 0001 86A0 hex)

Specify the frequency after acceleration/deceleration in pps.

S1+4	Lower word with number of output pulses
S1+5	Upper word with number of output pulses

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.

	15 0
F	Lower word with starting frequency
F+1	Upper word with starting frequency

0 to 100,000 pps
(0000 0000 to 0001 86A0 hex)

Specify the starting frequency in pps.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, M	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---			OK				
F											OK							

Flags

Name	Label	Operation
Error Flag	P_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), ORG(889) or IFEED(892). • 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. • ON if bit 14 (parameter change operation compensation) of M is ON and a constant speed cannot be achieved when changing the parameters. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

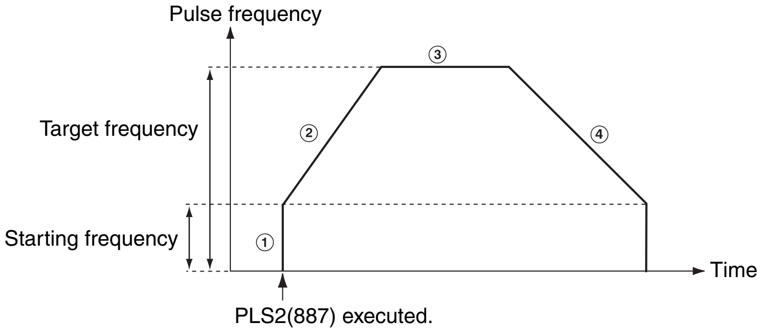
PLS2(887) starts pulse output on the port specified in P using the mode specified in M at the start frequency specified in F (1 in diagram).

The frequency is increased every pulse control period 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 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 for positioning only in independent mode.

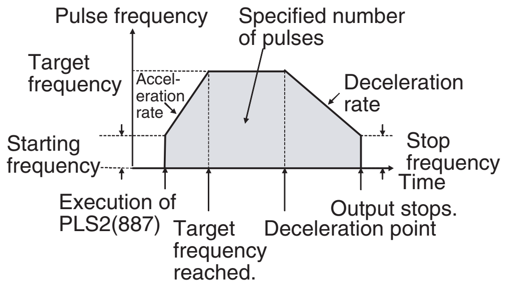
PLS2(887) can also be used to change settings during pulse output. With the CJ1M and CJ2M CPU Units, PLS2(887) can be executed during pulse output for ACC(888) in either independent or continuous mode, and during acceleration, constant speed, or deceleration. (See note.)

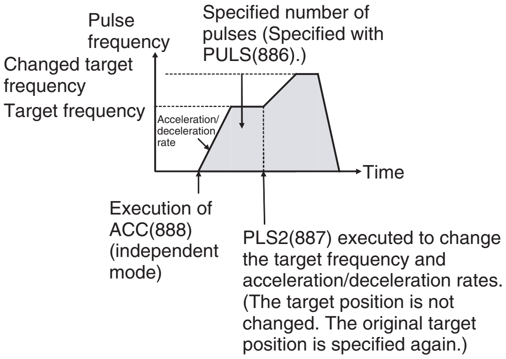
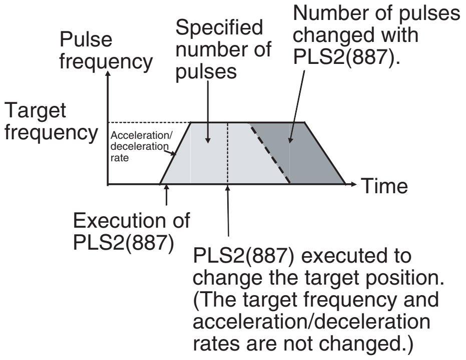
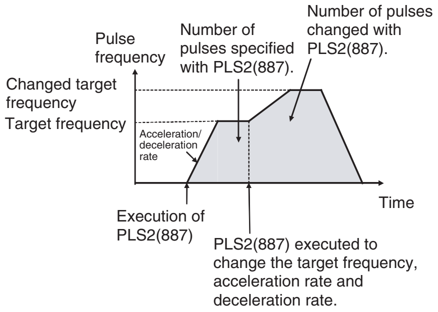
ACC(888) can also be executed during pulse output for PLS2(887) during acceleration, constant speed, or deceleration.

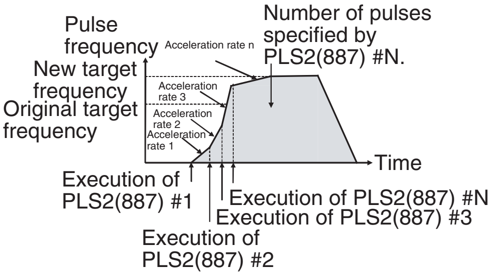
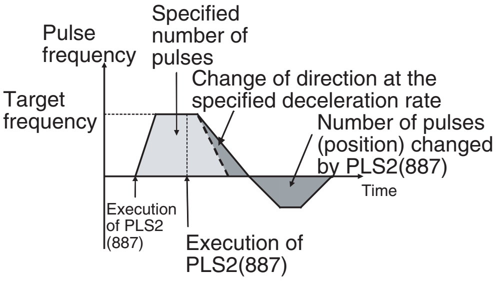
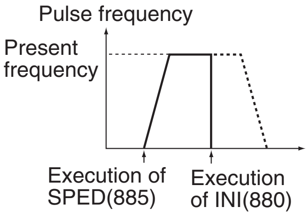
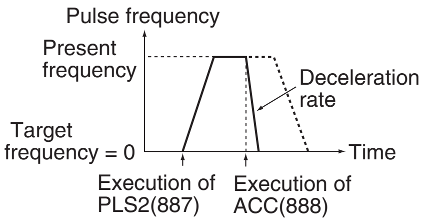
Note Executing PLS2(887) during speed control with ACC(888) (continuous mode) with the same target frequency as ACC(888) can be used to achieve interrupt feeding of a fixed distance. Acceleration will not be performed by PLS2(887) for this application, but if the acceleration rate is set to 0, the Error Flag will turn ON and PLS2(887) will not be executed. Always set the acceleration rate to a value other than 0.

Independent Mode Positioning

Note Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

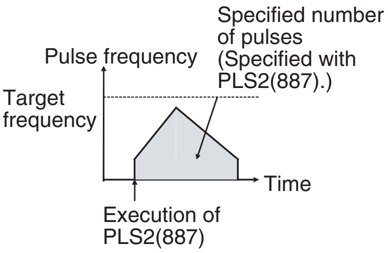
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	Complex trapezoidal control	Positioning with trapezoidal acceleration and deceleration (Separate rates used for acceleration and deceleration; starting speed) The number of pulses can be changed during positioning.		Accelerates and decelerates at a fixed rates. The pulse output is stopped when the specified number of pulses has been output. **Note** The target position (specified number of pulses) can be changed during positioning.	PLS2(887)

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
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** With the CJ1M or when operation compensation for parameter changes is not enabled with the CJ2M, an error will occur and the original operation will continue to the original target position if a constant speed cannot be maintained after changing the settings.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
	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** With the CJ1M or when operation compensation for parameter changes is not enabled with the CJ2M, an error will occur and the original operation will continue to the original target position if a constant speed cannot be maintained after changing the settings.	PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887) PLS2(887) ↓ PLS2(887)

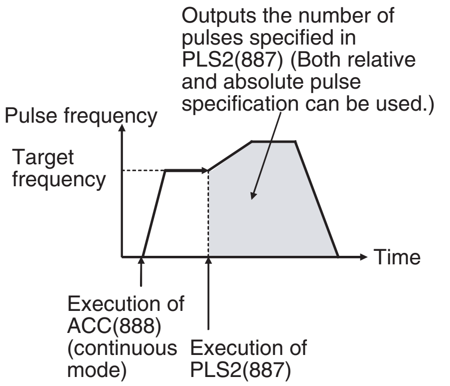
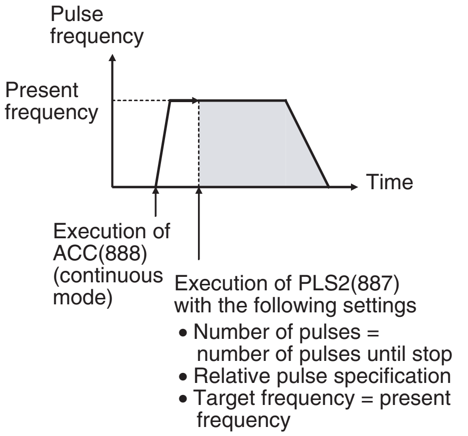
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Changing settings, continued	To change target position and speed smoothly, continued	Changing the acceleration and deceleration rates during positioning (multiple start function)		PLS2(887) can be executed during positioning (acceleration or deceleration) to change the acceleration rate or deceleration rate.	PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
					PLS2(887) ↓ 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)
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 pps)

Note Triangular Control

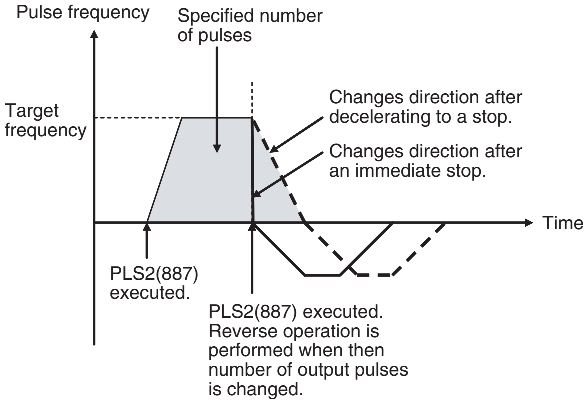
If the specified number of pulses is less than the number required to reach the target frequency and return to zero, the function will automatically reduce the acceleration/deceleration time and perform triangular control (acceleration and deceleration only.) An error will not occur.



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. **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.	ACC(888) (Continuous) ↓ PLS2(887)
Fixed distance feed interrupt			

Stopping Operation for Reversal Specifications



The stop method can be set to a deceleration stop or an immediate stop when changing the specified number of output pulses and reversing operation.

- When the stop operation for reversal specification (bit 15 of M) is set to 0 to set a deceleration stop, the direction will be changed using the specified deceleration rate. (This is indicated by the dotted line in the figure.)
- When the stop operation for reversal specification (bit 15 of M) is set to 1 to set an immediate stop, the direction will be changed after an immediate stop. (This is indicated by the solid line in the figure.)
- If the specified number of output pulses is less than the number of pulses that has been moved, reverse operation is performed.

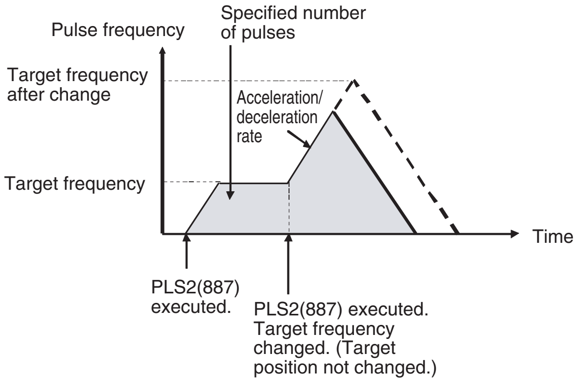
Note On the CJ1M-CPU2□, only a deceleration stop can be specified.

● Operation Compensation When Parameters Are Changed

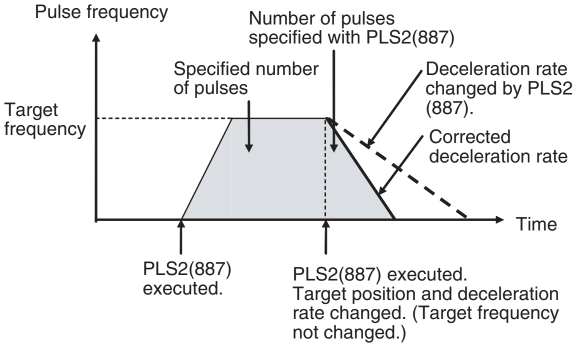
If a constant speed cannot be achieved or the target position is exceeded after parameters are changed, operation can be compensated and continued.

Note On the CJ1M-CPU2□, operation compensation cannot be used.

- Target Frequency Not Reached
  If operation compensation for parameter changes is enabled by setting bit 14 of M to 1, triangular control will be performed to approach the target frequency as much as possible. (The acceleration and deceleration rates will not be changed.)
  If operation compensation for parameter changes is disabled by setting bit 14 of M to 0, an instruction error will occur and PLS2(887) will not be executed.



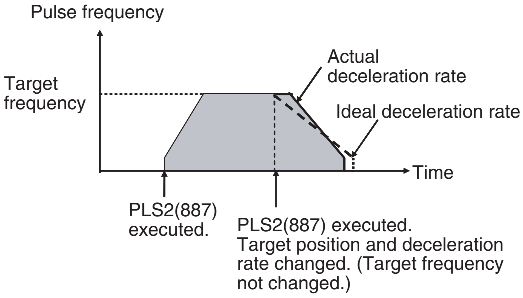
- Exceeding Target Position with Specified Deceleration Rate
  If operation compensation for parameter changes is enabled by setting bit 14 of M to 1, the deceleration rate will be changed to enable stopping at the target position.
  If operation compensation for parameter changes is disabled by setting bit 14 of M to 0, an instruction error will occur and PLS2(887) will not be executed.



Note 1 If the corrected deceleration rate would exceed 65,535, an immediate stop is performed.

2 If the corrected number of output pulses is smaller than the number of pulses that has been moved, reverse operation is performed instead of deceleration rate compensation. In this case, movement will stop and then movement to the target position will start again.

3 If the corrected deceleration rate is not an integer, it will be rounded up. The error that would occur from the decimal portion is compensated for at a constant speed and then deceleration is started.

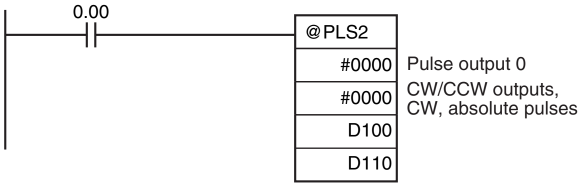


Precautions for Correct Use

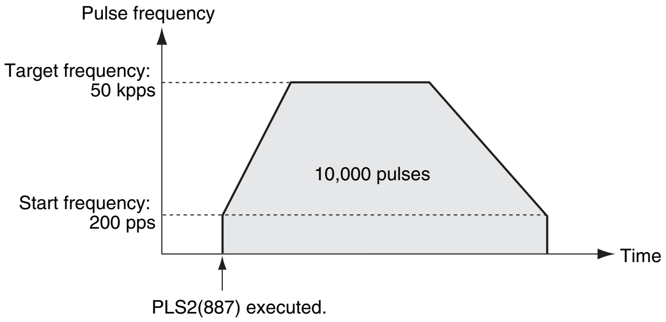
If PLS2(887) is executed within one cycle where pulse output stops (Output In-progress Flag would be ON), the system waits for pulse output to stop and then pulse output is restarted in the next cycle. If pulse output is stopped with INI(880), however, pulse output instructions executed within one cycle of stopped will not be executed. Wait for the Output In-progress Flag to turn OFF before executing the next instruction.

Example Programming

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 pps every 4 ms starting at 200 pps until the target speed of 50 kpps is reached. From the deceleration point, the pulse output is decelerated at a rate of 250 pps every 4 ms starting until the starting speed of at 200 pps is reached, at which point pulse output is stopped.

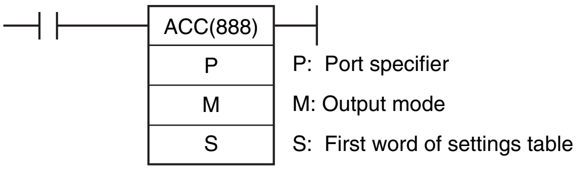


D100	01F4 hex	Acceleration rate: 500 pps/4 ms
D101	00FA hex	Deceleration rate: 250 pps/4 ms
D102	C350 hex	Target frequency: 50 kpps
D103	0000 hex	
D104	86A0 hex	Pulse output amount: 100,000 pulses
D105	0001 hex	
D110	00C8 hex	Start frequency: 200 pps
D111	0000 hex	



ACC

Instruction	Mnemonic	Variations	Function code	Function
ACCELERATION CONTROL	ACC	@ACC	888	ACC(888) outputs pulses to the specified output port at the specified frequency using the specified acceleration and deceleration rate.

	ACC
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

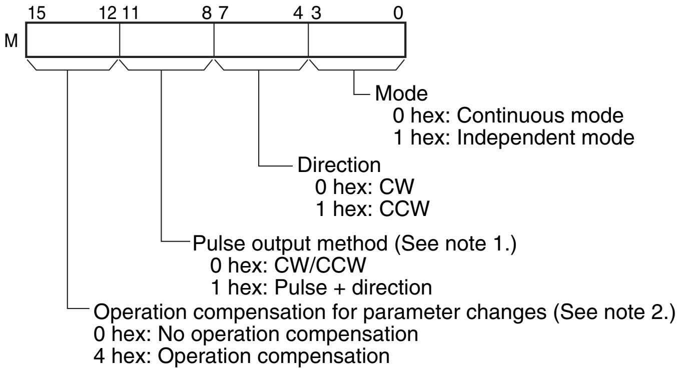
Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
M	Output mode	---	1
S	First word of settings table	WORD	3

P: Port Specifier

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)

M: Output Mode



Note 1 Use the same pulse output mode when using pulse outputs 0 and 1 at the same time and when using 2 and 3 at the same time.

2 If a constant speed cannot be achieved after parameters are changed, operation compensation can be set to continue operation. (CJ2M Only)

S: First Word of Settings Table

15 0
S Acceleration/deceleration rate

CJ1M CPU Units, no unit version
1 to 2000 pps (#0001 to 07D0)
CJ1M CPU Unit with unit version 2.0 or later or CJ2M CPU Unit
1 to 65535 pps (#0001 to FFFF)

Specify the increase or decrease in the frequency per pulse control period (4 ms).

S+1 Lower word with target frequency
S+2 Upper word with target frequency

0 to 100,000 pps
(0000 0000 to 0001 86A0 hex)

Specify the frequency after acceleration or deceleration in pps.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, M	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---			OK				

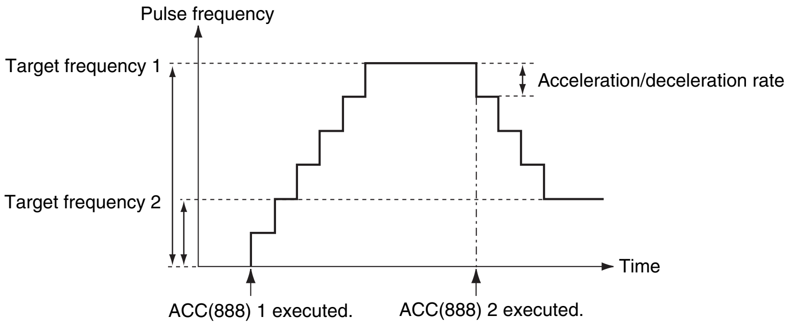
Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the specified range for P, M, or S is exceeded. • ON if pulses are being output using ORG(889) or IFEED(892) 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. • ON if bit 14 (parameter change operation compensation) of M is ON to specify not using parameter change operation compensation and a constant speed cannot be achieved when changing the parameters. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

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+1 and S+2 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 continuous mode, pulse output will continue until stopped from the program.

In independent mode, pulse output will stop automatically when the number of pulses set with PULS(886) in advance have been output.

An error will occur if the mode is changed between independent and continuous mode while pulses are being output.

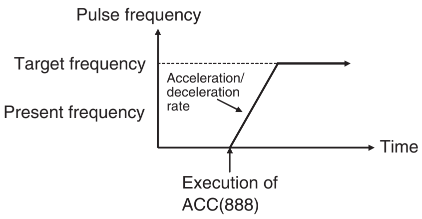
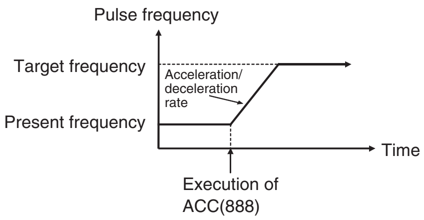
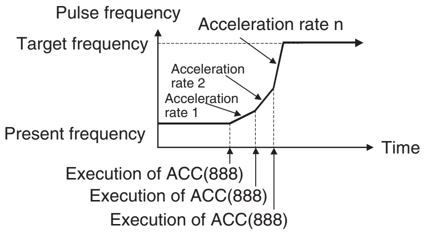
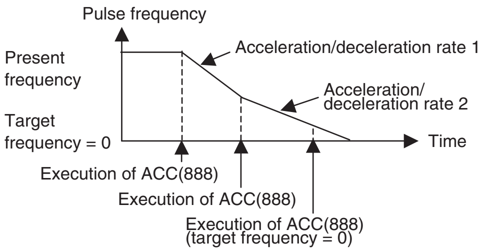
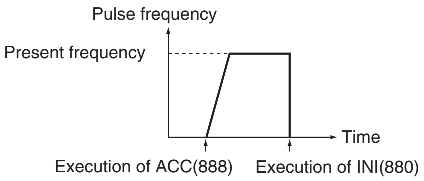
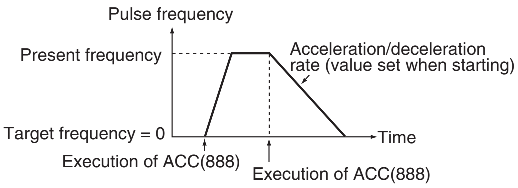
ACC(888) can also be used to change settings during pulse output. ACC(888) in independent mode can also be executed during pulse output for PLS2(887) during acceleration, constant speed, or deceleration. With the CJ1M and CJ2M CPU Units, PLS2(887) can be executed during pulse output for ACC(888) in either independent or continuous mode, and during acceleration, constant speed, or deceleration. (See note 1.)

Note 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.

● 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)
		Decelerating to a stop		The acceleration/ deceleration rate is changed while decelerating. **Note** If the target frequency is set to 0 pps, the current acceleration/ deceleration rate will be used.	ACC(888) (Continuous) ↓ ACC(888) (Continuous) ↓ ACC(888) (Continuous, target frequency of 0 pps)
Stopping pulse output	To stop pulse output	Immediate stop		Immediately stops pulse output.	ACC(888) (Continuous) ↓ INI(880) (Continuous)
	To stop pulse output smoothly	Decelerating to a stop		Decelerated pulse output to a stop. **Note** If the target frequency of the second ACC(888) instruction is 0 pps, the acceleration/deceleration rate from the first ACC(888) instruction will be used.	ACC(888) (Continuous) ↓ ACC(888) (Continuous, target frequency of 0 pps)

● Independent Mode Positioning

When independent mode operation is started, pulse output will be continued until the specified number of pulses has been output.

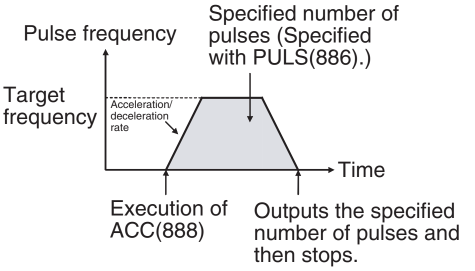
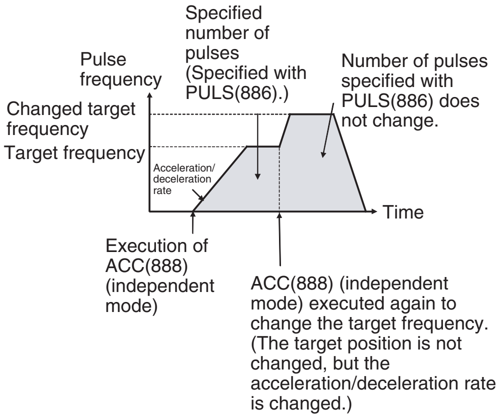
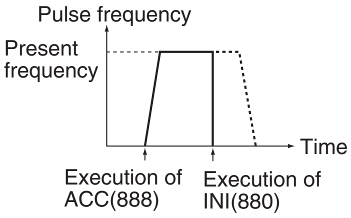
The deceleration point is calculated from the number of output pulses and deceleration rate set in S and when that point is reached, the frequency is decreased every pulse control period at the deceleration rate specified in S until the specified number of points has been output, at which point pulse output is stopped.

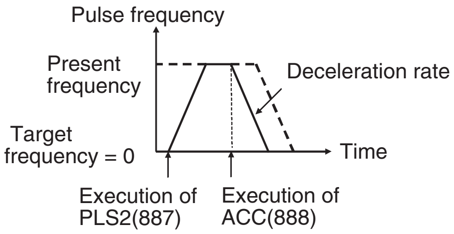
Note 1 Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

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.

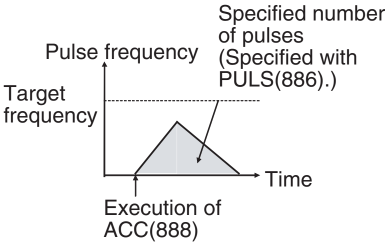
4 The direction set in the ACC(888) operand will be ignored if the number of pulses is set with PULS(881) as an absolute value.

Opera-tion	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting and stopping 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 decelerates to a stop 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) ↓ ACC(888) (Independent) PLS2(887) ↓ ACC(888) (Independent)
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)

Opera-tion	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output, continued	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.	PULS(886) ↓ ACC(888) (Independent) ↓ ACC(888) (Independent, independent, target frequency of 0 pps)
					PLS2(887) ↓ ACC(888) (Independent, target frequency of 0 pps)

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.

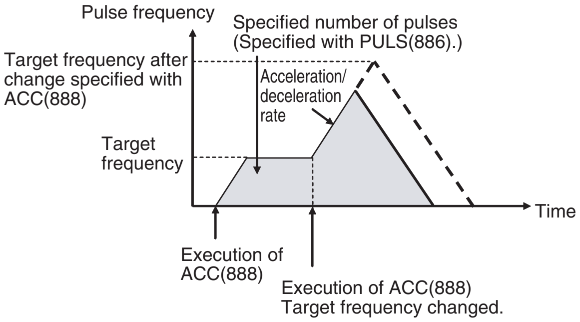


● Operation Compensation When Parameters Are Changed

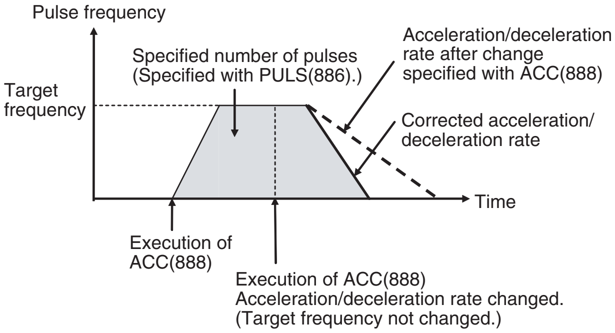
If a constant speed cannot be achieved or the target position is exceeded after parameters are changed in independent mode, operation can be compensated and continued.

Note On the CJ1M-CPU2□, operation compensation cannot be used.

- Target Frequency Not Reached
  If operation compensation for parameter changes is enabled by setting bit 14 of M to 1, triangular control will be performed to approach the target frequency as much as possible.
  If operation compensation for parameter changes is disabled by setting bit 14 of M to 0, an instruction error will occur and ACC(888) will not be executed.

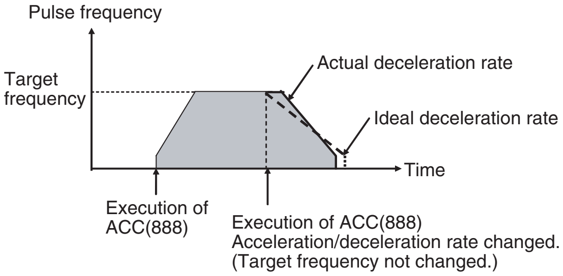


- Exceeding Target Position with Specified Acceleration/Deceleration Rate
  If operation compensation for parameter changes is enabled by setting bit 14 of M to 1, the deceleration rate will be changed to enable stopping at the target position.
  If operation compensation for parameter changes is disabled by setting bit 14 of M to 0, an instruction error will occur and ACC(888) will not be executed.



Note 1 If the corrected deceleration rate would exceed 65,535, an immediate stop is performed.

2 If the corrected deceleration rate is not an integer, it will be rounded up. The error that would occur from the decimal portion is compensated for at a constant speed and then deceleration is started.



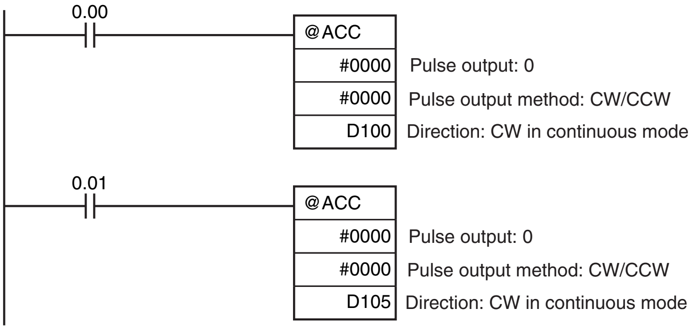
Precautions for Correct Use

If ACC(888) or PLS2(887) is executed within one cycle where pulse output stops (Output In-progress Flag would be ON), the system waits for pulse output to stop and then pulse output is restarted in the next cycle. If pulse output is stopped with INI(880), however, pulse output instructions executed within one cycle of stopped will not be executed. Wait for the Output In-progress Flag to turn OFF before executing the next instruction.

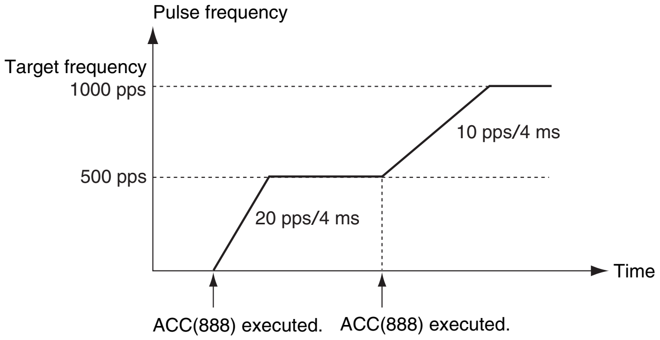
If pulse output that is being executed for SPED(885) is stopped by setting the target frequency to 0 pps with SPED(885) or ACC(888), operation will be the same as when pulse output is stopped with INI(880). Wait for the Output In-progress Flag to turn OFF before executing the next positioning instruction.

Example Programming

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 pps every 4 ms until the target frequency of 500 pps is reached. When CIO 0.01 turns ON, ACC(888) changes to an acceleration rate of 10 pps every 4 ms until the target frequency of 1,000 pps is reached.

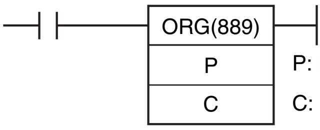


D100	#0014	Acceleration/deceleration rate: 20 pps
D101	#01F4	Target frequency: 500 pps
D102	#0000	
D105	#000A	Acceleration/deceleration rate: 10 pps
D106	#03E8	Target frequency: 1,000 pps
D107	#0000	



ORG

Instruction	Mnemonic	Variations	Function code	Function
ORIGIN SEARCH	ORG	@ORG	889	ORG(889) performs an origin search or origin return operation.

Symbol	ORG
	 ORG(889) P: Port specifier C: Control data

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

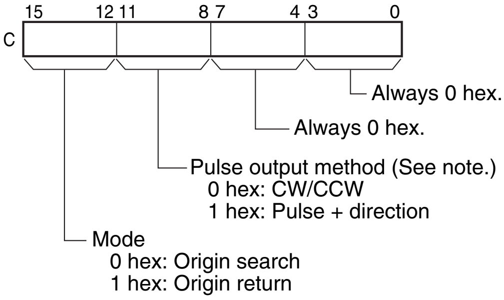
Operands

Operand	Description	Data type	Size
P	Port specifier	---	1
C	Control data	---	1

P: Port Specifier

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0002 hex	Pulse output 2 (CJ2M only)
0003 hex	Pulse output 3 (CJ2M only)

C: Control Data



Always 0 hex.

Always 0 hex.

Pulse output method (See note.)
0 hex: CW/CCW
1 hex: Pulse + direction

Mode
0 hex: Origin search
1 hex: Origin return

Note Use the same pulse output method when using both pulse outputs 0 and 1.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P,C	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_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), PLS2(887) or IFEED(892). • 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 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 an origin return is specified and the origin has not been established. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

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.

Origin search	Origin return
• Origin Search Function Enable/Disable • Origin Search Operating Mode • Origin Search Operation Setting • Origin Detection Method • Origin Search Direction Setting • Origin Search/Return Initial Speed • Origin Search High Speed • Origin Search Proximity Speed • Origin Search Acceleration Rate • Origin Search Deceleration Rate • Origin Compensation • Origin Proximity Input Signal Type • Origin Input Signal Type • Limit Input Signal Type • Positioning Monitor Time	• Origin Return Target Speed • Origin Search/Return Initial Speed • Origin Return Acceleration Rate • Origin Return Deceleration Rate

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).

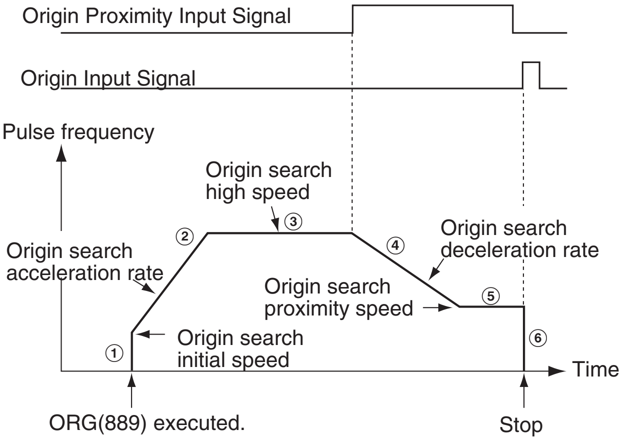
Pulse output is accelerated to the Origin Search High Speed using the Origin Search Acceleration Rate (2 in diagram).

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).

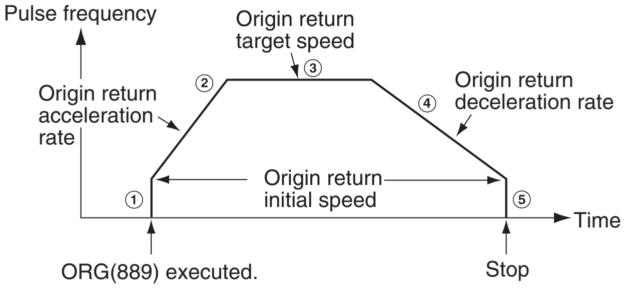
Pulse output is stopped when the Origin Input Signal turns ON (6 in diagram).

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.

● 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).

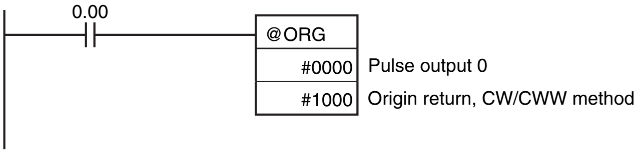
The deceleration point is calculated from the number of pulses remaining to the origin and the deceleration rate and when that point is reached, the pulse output is decelerated at the Origin Return Deceleration Rate (4 in diagram) until the Origin Return Start Speed is reached, at which point pulse output is stopped at the origin (5 in diagram).

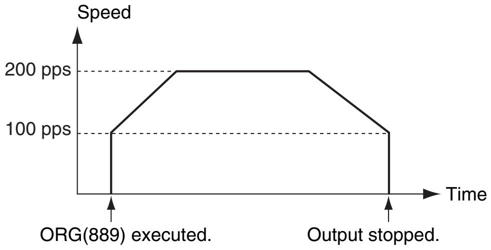
Hint

For the CJ2M, INI(880) can be executed to change some of the origin search/return parameters during operation, such as the Starting Speed, Target Speed, or Acceleration/Deceleration Rate, by setting C to 0005 hex.

Example Programming

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 pps/4 ms.



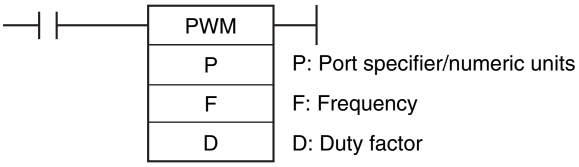


The PLC Setup parameters are as follows:

Parameter	Setting
Pulse Output 0 Starting Speed for Origin Search and Origin Return	100 pps
Pulse Output 0 Origin Return Target Speed	200 pps
Pulse Output 0 Origin Return Acceleration Rate	50 pps/4 ms
Pulse Output 0 Origin Return Deceleration Rate	50 pps/4 ms

PWM

Instruction	Mnemonic	Variations	Fun No.	Function
PWM outputs	PWM	@PWM	891	PWM(891) is used to output pulses with a variable duty factor.

	PWM
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
P	Port specifier/numeric units	---	1
F	Frequency	UNIT	1
D	Duty factor	UNIT	1

P: Port specifier/numeric units

Set value	Frequency unit	Duty ratio unit	Setting support	
			CJ1M-CPU2□	CJ2M+MD21□
#000□	0.1 Hz	1%	OK	OK
#100□	0.1 Hz	0.1%	OK	OK
#110□	1 Hz	0.1%	Not allowed.	OK

		Setting support		
		CJ1M-CPU21	CJ1M-CPU22/23	CJ2M+MD21□
Port number	0	OK	OK	OK
	1	Not allowed.	OK	OK
	2	Not allowed.	Not allowed.	OK
	3	Not allowed.	Not allowed.	OK

F: Frequency

Numeric unit	Setting range	Setting support	
		CJ1M-CPU2□	CJ2M+MD21□
0.1 Hz	0.1 to 6,553.5 Hz 1 to 65,535 (0001 to FFFF hex)	OK	OK
1 Hz	1 to 32,800 Hz 1 to 32,800 (0001 to 8020 hex)	Not allowed.	OK

D: Duty factor

Numeric unit	Setting range	Setting support	
		CJ1M-CPU2□ (pre-version 2.0)	CJ1M-CPU2□ with unit version 2.0 or later or CJ2M + MD21□
1%	0% to 100% 0 to 100 (0000 to 0064 hex)	OK	OK
0.1%	0.0% to 100.0% 0 to 1,000 (0000 to 03E8 hex)	Not allowed.	OK

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
F, D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the specified range for P, F, or D is exceeded. • ON if the origin search function is assigned to the PWM output terminal in the PLC Setup. • ON if PWM(891) is executed in an interrupt task for the same port number when PWM(891) is already being executed. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

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 PWM output to change the duty factor without stopping PWM output. Any attempts to change the frequency will be ignored. PWM 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 PWM output will continue until stopped using one of the following occurs:

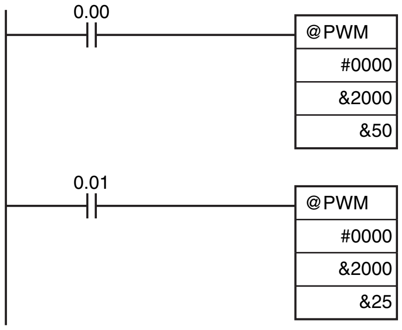
(1) INI(880) is executed to stop it (C = 0003 hex: stop PWM output).

(2) The CPU Unit is switched to PROGRAM mode.

Example Programming

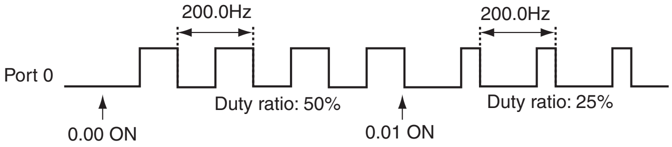
When CIO 0.00 turns ON in the following programming example, PWM(891) starts pulse output from PWM output 0 at 200 Hz with a duty factor of 50%.

When CIO 0.01 turns ON, the duty factor is changed to 25% from the next output pulse.



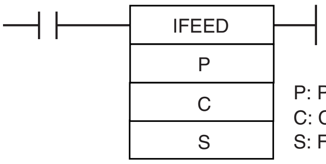
PWM output
Port 0, Frequency in 0.1-Hz increments and duty factor in 1% increments
Frequency: 200.0 Hz
Duty ratio: 50%

PWM output
Port 0, Frequency in 0.1-Hz increments and duty factor in 1% increments
Frequency: 200.0 Hz
Duty ratio: 25%



IFEED

Instruction	Mnemonic	Variations	Function code	Function
INTERRUPT FEEDING	IFEED	@IFEED	892	IFEED(892) uses an input interrupt as a trigger to switch from speed control to position control and move the specified number of pulses. IFEED(892) is supported only by the CJ2M.

	PWM
Symbol	 P: Port Specifier C: Control Data S: First word of settings table

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

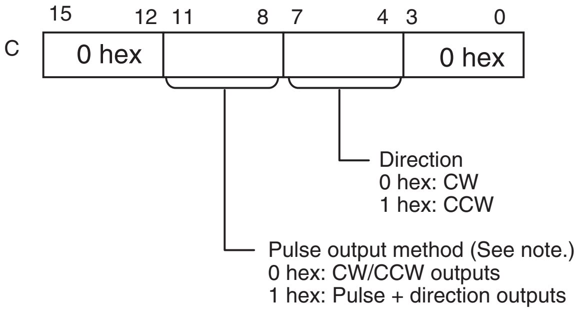
Operand	Description	Data type	Size
P	Port specifier	---	1
C	Control word	---	1
S	First word of settings table	WORD	6

P: Port Specifier

P	
0000 hex	Input interrupt 0 – Pulse output 0
0001 hex	Input interrupt 1 – Pulse output 1
0002 hex	Input interrupt 4 – Pulse output 2
0003 hex	Input interrupt 5 – Pulse output 3

Note The input interrupt and pulse output combinations given above must be used. They cannot be changed.

C: Control Data



Direction
0 hex: CW
1 hex: CCW

Pulse output method (See note.)
0 hex: CW/CCW outputs
1 hex: Pulse + direction outputs

Note: Use the same pulse output method when using pulse outputs 0 and 1 at the same time and when using 2 and 3 at the same time.

S: First word of settings table

S	Acceleration Rate	1 to 65,535 pps/4 ms (0001 to FFFF hex)
S+1	Deceleration Rate	1 to 65,535 pps/4 ms (0001 to FFFF hex)
S+2	Lower word of target frequency	0 to 100 kpps (0000 0000 to 0001 86A0 hex)
S+3	Upper word of target frequency	
S+4	Lower word with number of pulses	Number of output pulses: 0 to 2,147,483,647 (0000 0000 to 7FFF FFFF hex)
S+5	Upper word with number of pulses	

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
P, C	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the specified range for P, C, or S is exceeded. • ON if an input that is being used for an input interrupt is not set as an input interrupt in the PLC Setup. • ON if IFEED(892) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. • ON if pulse output is already being performed for the pulse output port. • ON if using interrupts is enabled for the specified input interrupt. • ON if the number of output pulses specified in S is less than the number of pulses required to decelerate to a stop for the specified operation. • For a CJ2M CPU Unit, ON for any function that uses I/O on the Pulse I/O Module even if a Pulse I/O Module is not mounted. • OFF in all other cases.

Function

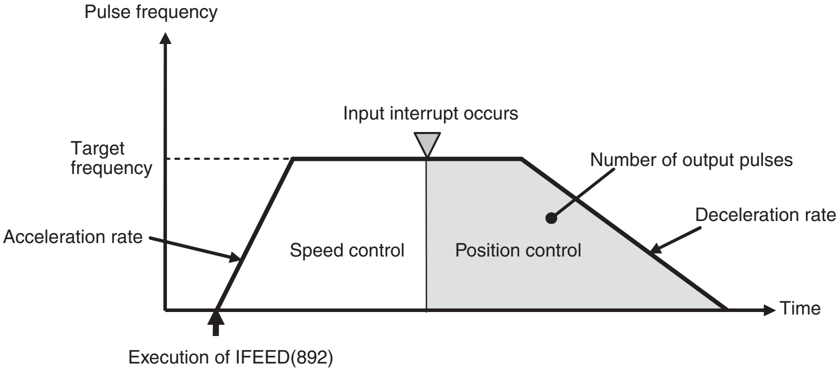
IFEED(892) starts pulse output from the port specified in P using the method specified in C. Movement accelerates at the acceleration rate specified in S to the target frequency specified in S and speed control is performed in continuous mode. Then, when the input interrupt specified in C occurs (see note), the system switches to position control, outputs the number of pulses specified by S and decelerates at the deceleration rate specified by S.

Note Direct mode interrupts for the interrupt inputs are enabled by IFEED(892). It is not necessary to execute MSKS(690). Even if an interrupt task exists, it will not be executed. However, to create an input interrupt when the interrupt input turns OFF, execute MSKS(690) before IFEED(892) to specify downward differentiation. Unless MSKS(690) is used to specify downward differentiation, an input interrupt will be generated for IFEED(892) when the interrupt input turns ON.

IFEED(892) performs control by combining a specific pulse output with an input interrupt. It does not use an interrupt task. Rather, interrupt feeding is set and executed separately for each IFEED(892) instruction.

This achieves faster interrupt response than starting an interrupt task and executing PLS2(887) in the interrupt task. The input interrupt and pulse output combinations given above must be used. They cannot be changed. Once pulse output has been started with IFEED(892), no other pulse output instructions except for INI(880) can be executed, and INI(880) can be used only to stop pulse output. If INI(880) is executed to stop pulse output, pulse output will be stopped and the input interrupt will be masked. If IFEED(892) is executed again, pulse output will be started from the beginning.

To use other combinations of pulse outputs and input interrupts or to change settings during pulse output, use ACC(888) and PLS2(887).



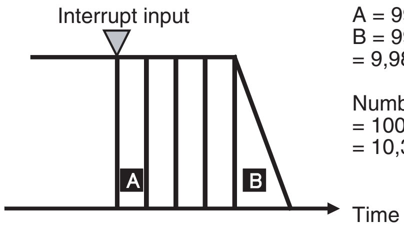
Precautions for Correct Use

An instruction error will occur if a constant speed cannot be achieved for the number of pulses specified by S. If that occurs, set the number of output pulses so that it is greater than the number of pulses found from the target frequency and deceleration rate using the following formula.

Number of output pulses for no error = Number of pulses in 1 pulse control cycle at the target frequency* × 4 + Number of pulses required to decelerate from the target frequency*

* Round up below the decimal point.

Example: Target frequency: 99,900 pps
Deceleration rate: 2,000 pps/4 ms
Pulse control cycle: 1 ms

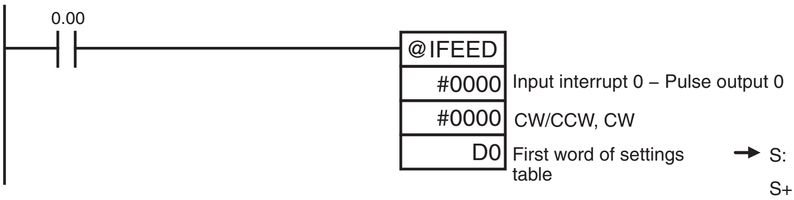


A = 99,900 pps × 0.001 s = 99.9 pulses --> 100 pulses
B = 99,900 pps × (99,900 pps/2,000 pps × 0.004 s)/2
= 9,980.01 pulses --> 9,981 pulses

Number of output pulses for no error = A × 4 + B
= 100 × 4 + 9,981
= 10,381 pulses

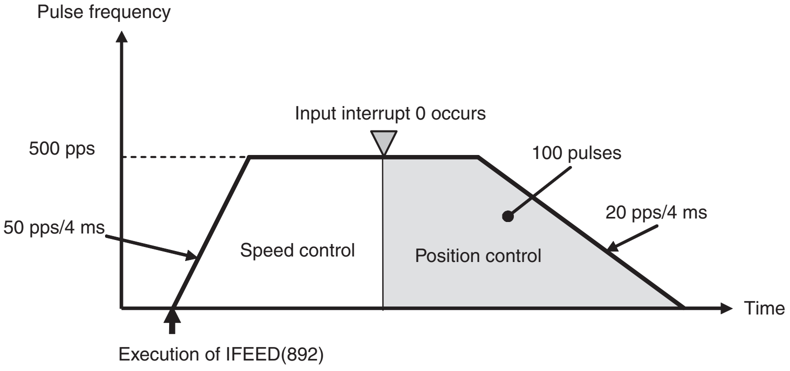
Example Programming

When CIO 0.00 turns ON, pulse output from pulse output 0 is started. The system accelerates at 50 pps/4 ms to a target frequency of 500 pps and then performs speed control in continuous mode. When input interrupt 0 occurs, the system switches to position control and then decelerates at 20 pps/4 ms to stop after outputting the specified number of pulses.



Input interrupt 0 – Pulse output 0
CW/CCW, CW
First word of settings table

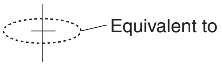
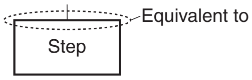
S:	D0	#0032	Acceleration rate: 50 pps/4 ms
S+1:	D1	#0014	Deceleration rate: 20 pps/4 ms
S+2:	D2	#01F4	Target frequency: 500 pps
S+3:	D3	#0000	
S+4:	D4	#0064	Number of output pulses: 100 pulses
S+5:	D5	#0000	

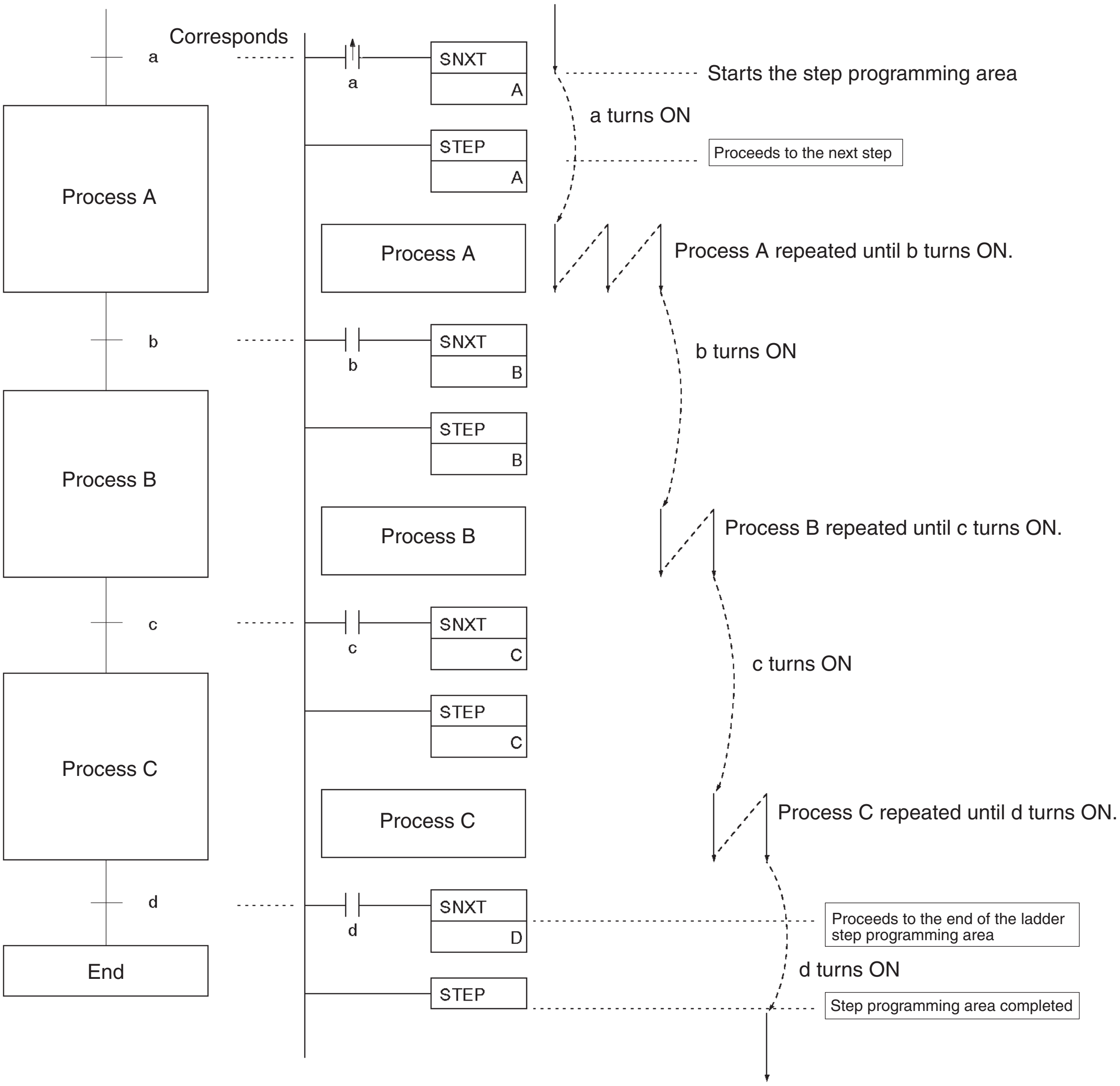


Step Instructions

Step Instructions

In CS/CJ-series PLCs, STEP(008)/SNXT(009) can be used together to create step programs.

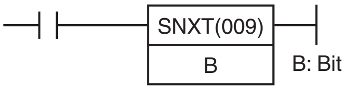
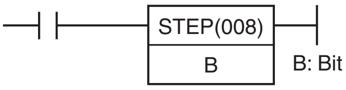
Instruction	Operation	Diagram
SNXT(009): STEP START	Controls progression to the next step of the program.	Step Ladder section start instruction 
STEP(008): STEP DEFINE	Indicates the start of a step. Repeats the same step program until the conditions for progression to the next step are established.	Step Ladder section start instruction 



Note Work bits are used as the control bits for A, B, C and D.

SNXT/STEP

Instruction	Mnemonic	Variations	Function code	Function
STEP START	SNXT	---	009	SNXT(009) is used to control progression of step execution in the step program area.
STEP DEFINE	STEP	---	008	STEP(008) is used to define the beginning and the end of the step program area.

	SNXT	STEP
Symbol		When defining the beginning of a step, a control bit is specified as follows:  When defining the end of a step, a control bit is not specified as follows: 

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	OK	Not allowed	Not allowed	Not allowed

Operands

Operand	Description	Data type	Size
B	Bit	---	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	---	OK	---	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation	
		SNXT	STEP
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.	• 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.

Function

● SNXT(009)

SNXT(009) is used in the following three ways:
1.To start step programming execution.
2. To proceed to the next step control bit.
3. To end step programming execution.

The step program 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 program 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 program 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 program 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. Starts a specific step.
2. Ends the step program 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
TIM, TIMX(551), TIMH(015), TIMHX(551), TMHH(540), TIMHHX(552), TIML(542), and TIMLX(553)	PV	0000 hex (reset)
	Completion Flag	OFF (reset)
TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557)*1	Cannot be read.	
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).

*1 CJ1-H-R and CJ2 CPU Units only

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.

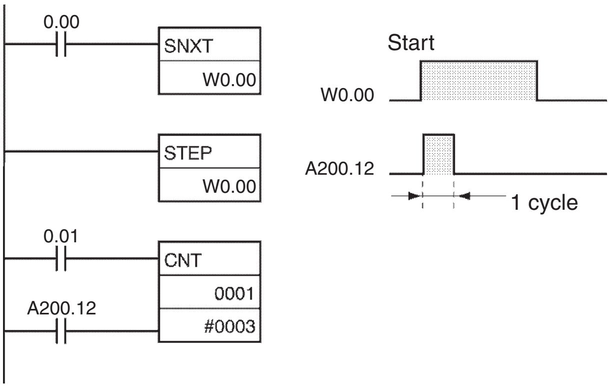
Ending the Step Program Area

STEP(008) is placed at the end of the step program 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).

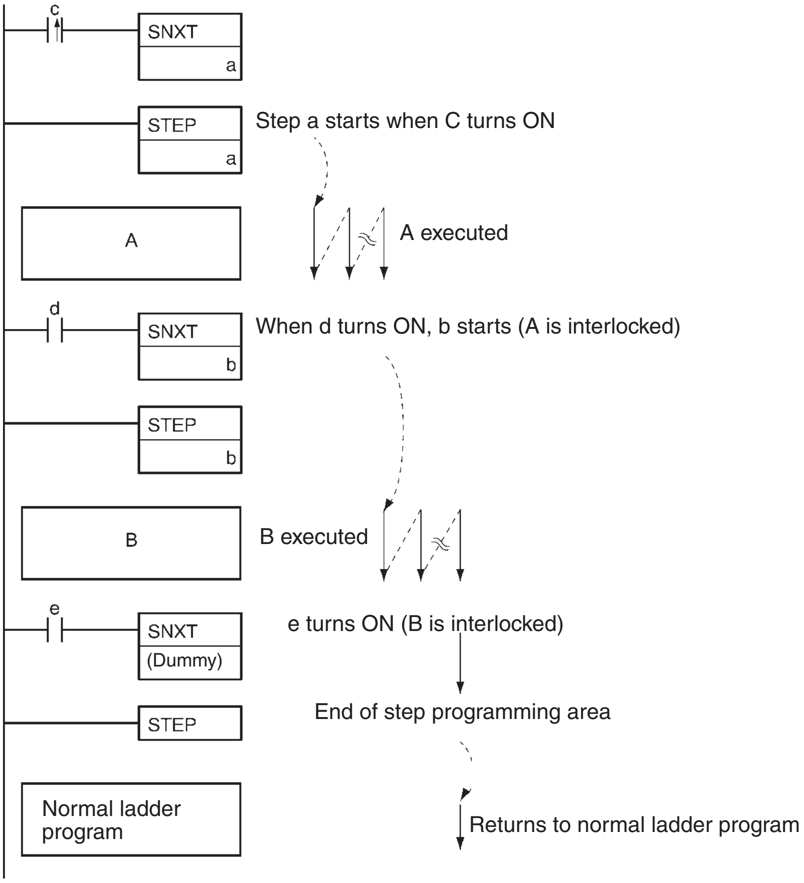
Hint

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.



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.



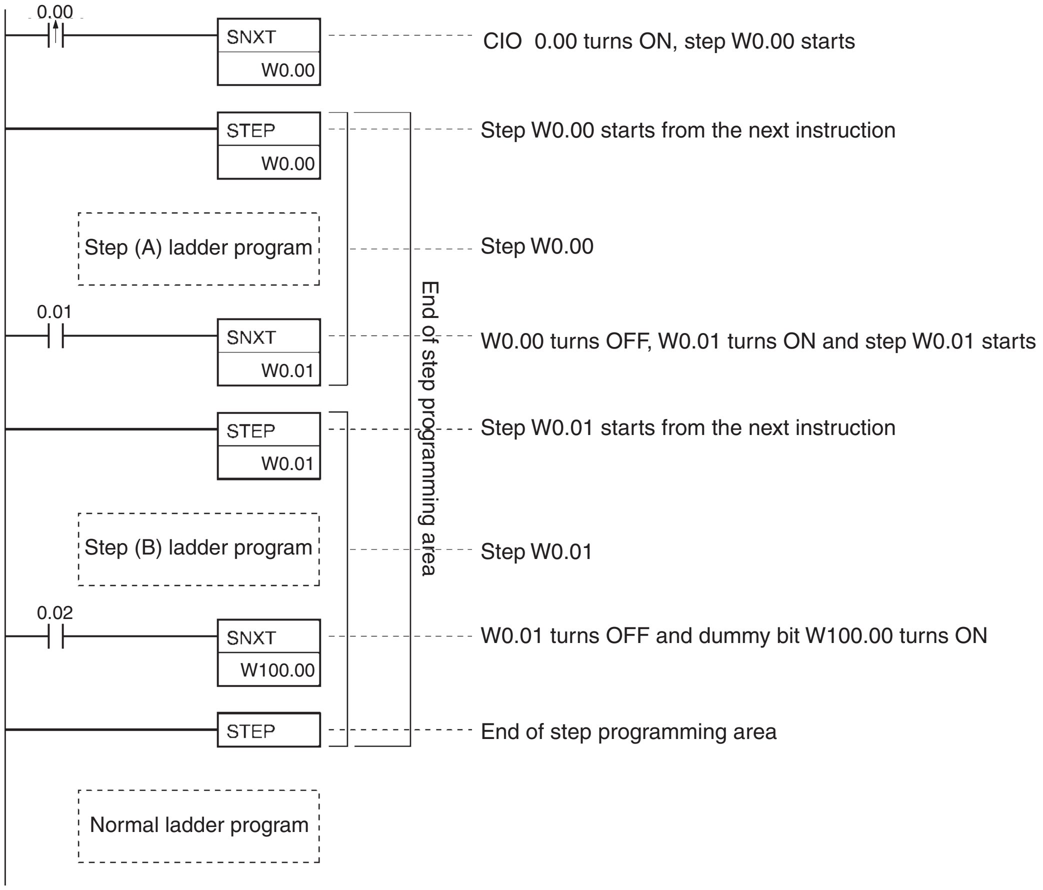
Precaution

- 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.
- SNXT(009) is executed when the execution condition is ON.
- Input SNXT(009) at the end of the step program 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 program area, the corresponding step will be started when SNXT(009) is executed.

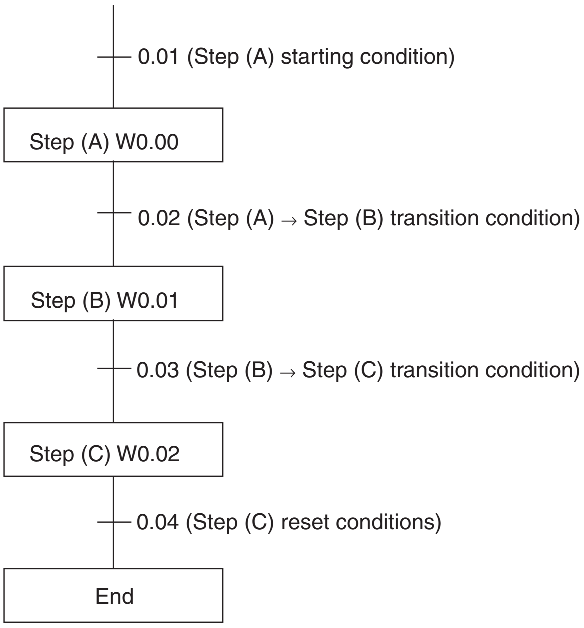
- STEP(008) and SNXT(009) cannot be used inside of subroutines, interrupt programs, or block programs.
- The instructions that cannot be used within step programs are listed in the following table.

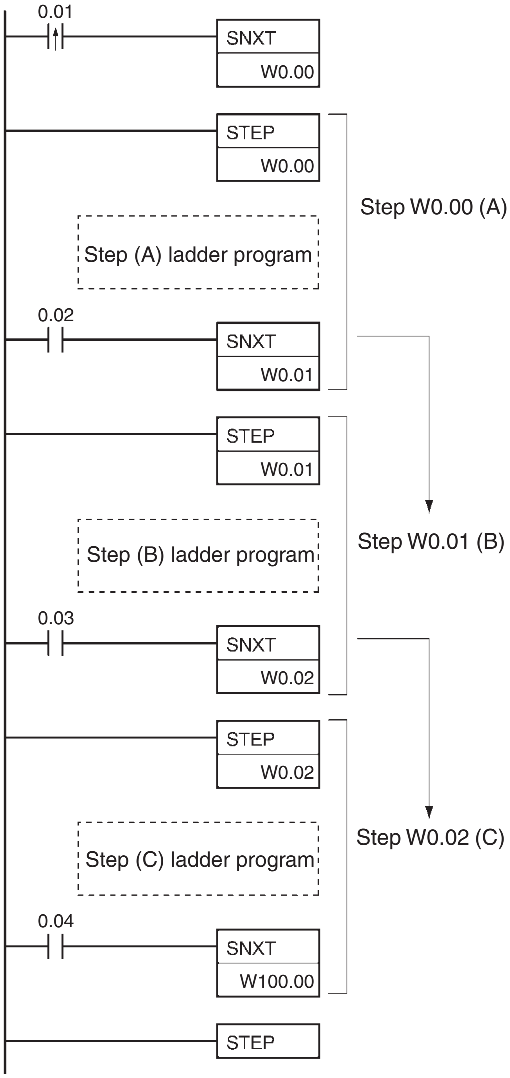
Function	Mnemonic	Name
Sequence Control Instructions	END(001)	END
	IL(002)	INTERLOCK
	ILC(003)	INTERLOCK CLEAR
	JMP(004)	JUMP
	JME(005)	JUMP END
	CJP(510)	CONDITIONAL JUMP
	CJPN(511)	CONDITIONAL JUMP NOT
	JMP0(515)	MULTIPLE JUMP
	JME0(516)	MULTIPLE JUMP END
Subroutine Instructions	SBN(092)	SUBROUTINE ENTRY
	RET(093)	SUBROUTINE RETURN

Example Programming

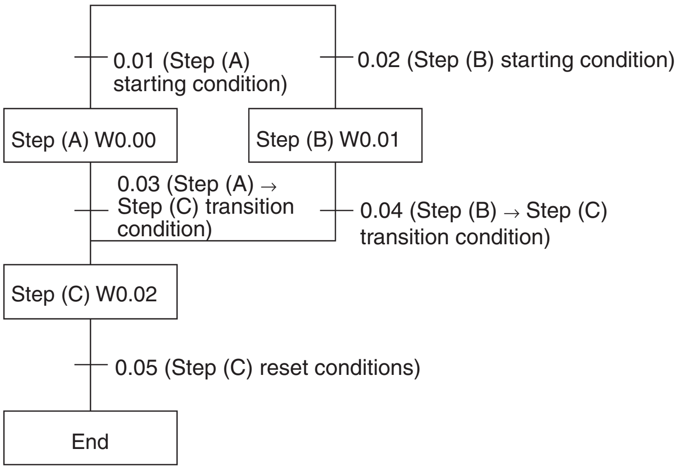


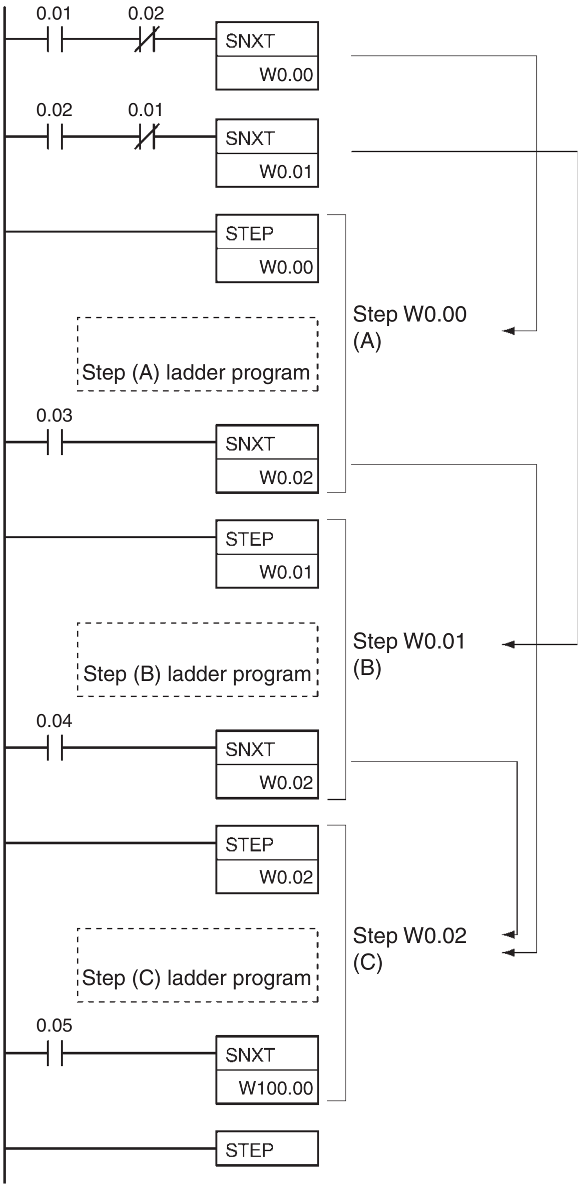
(1) Sequential Control





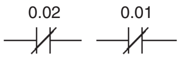
(2) Branching Control



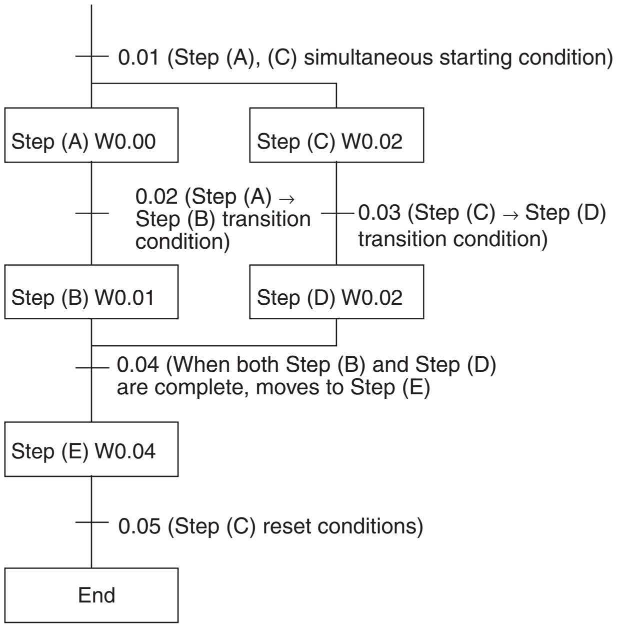


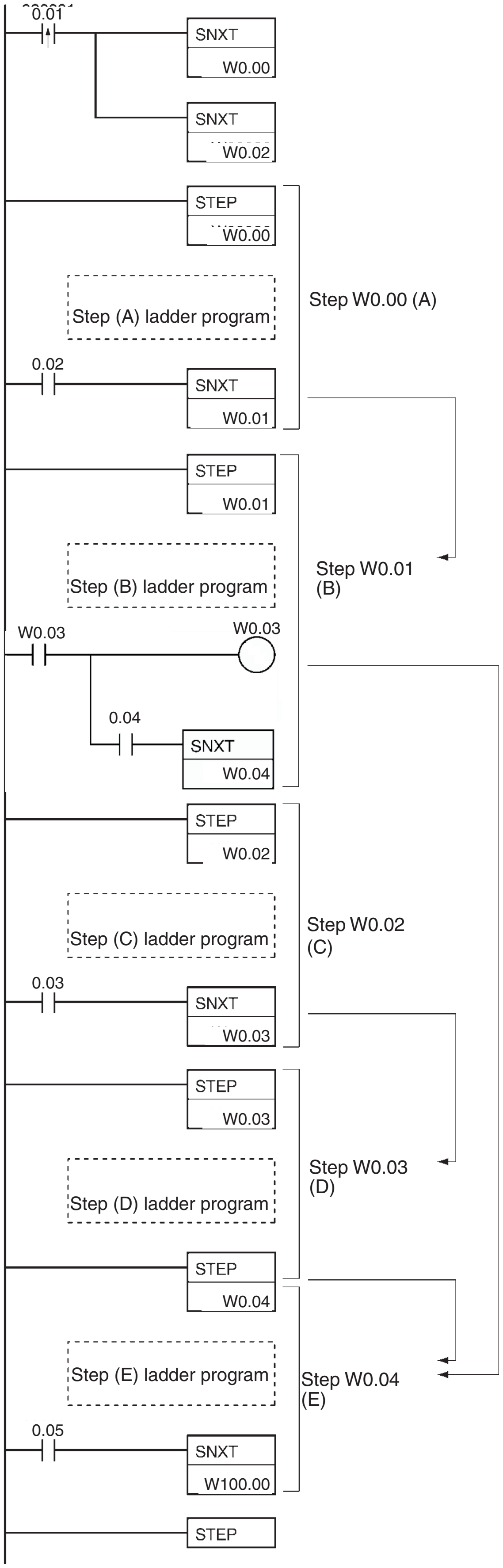
● Additional Information:

- 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.
- 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.



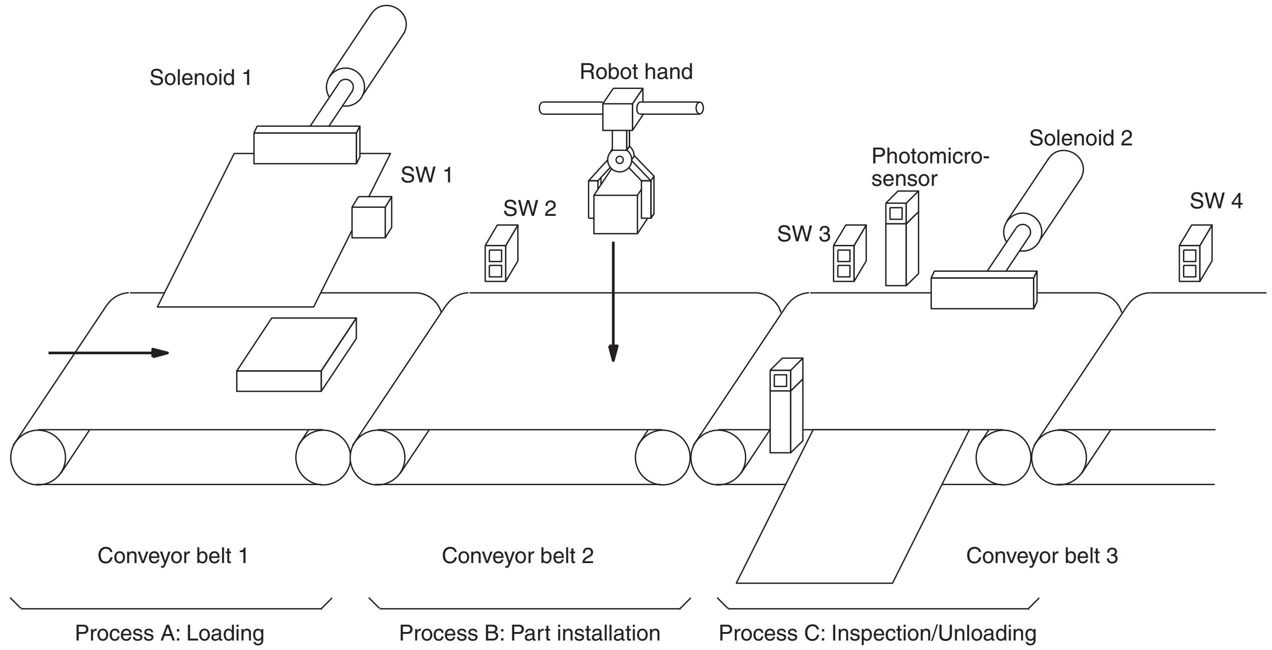
(3) Parallel Control

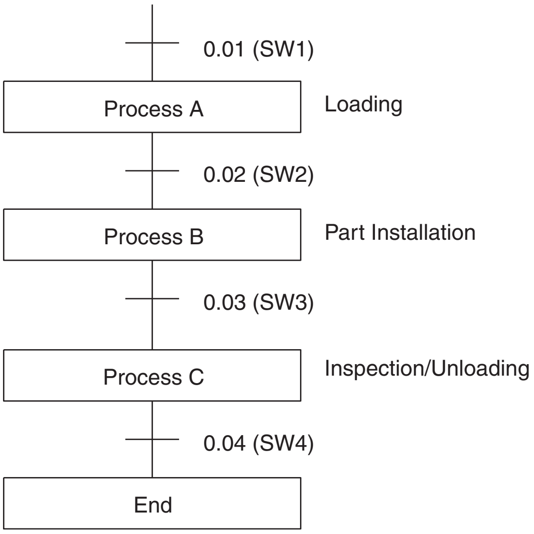


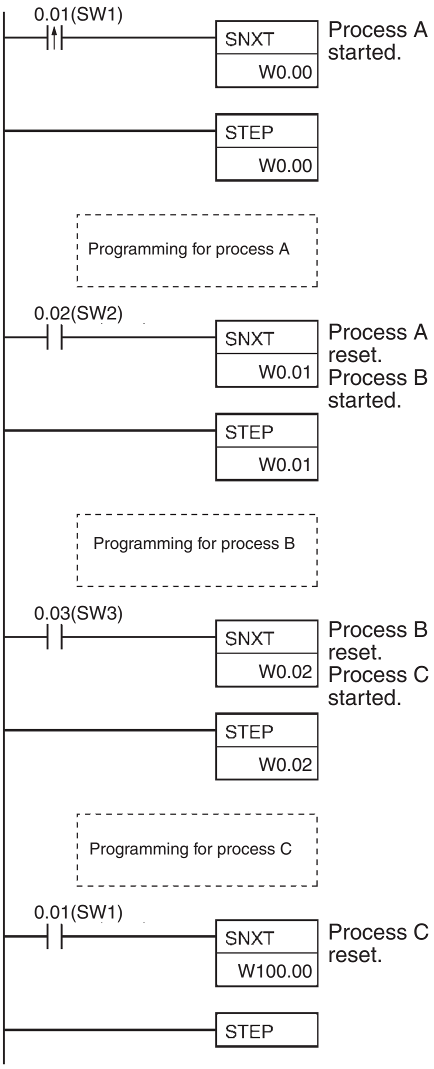


Application Examples

(1) Sequential Execution







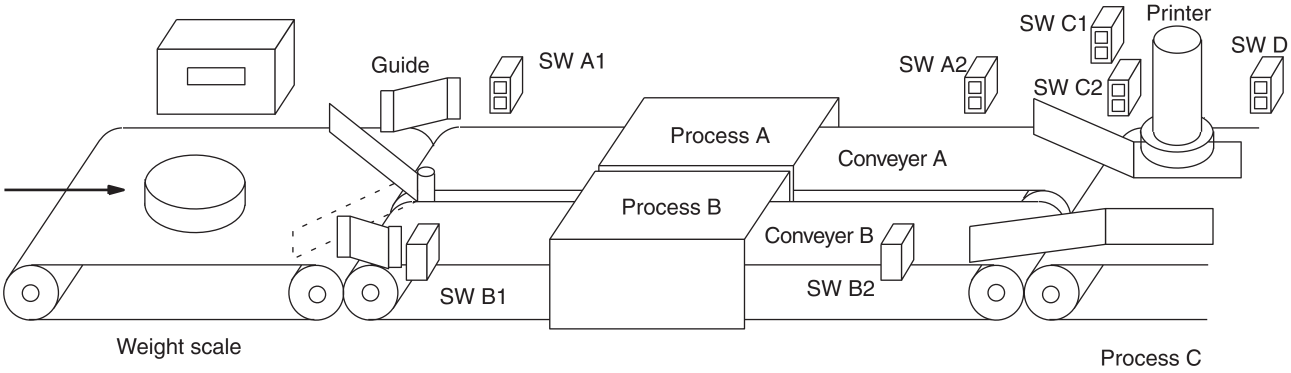
Explanation of operation

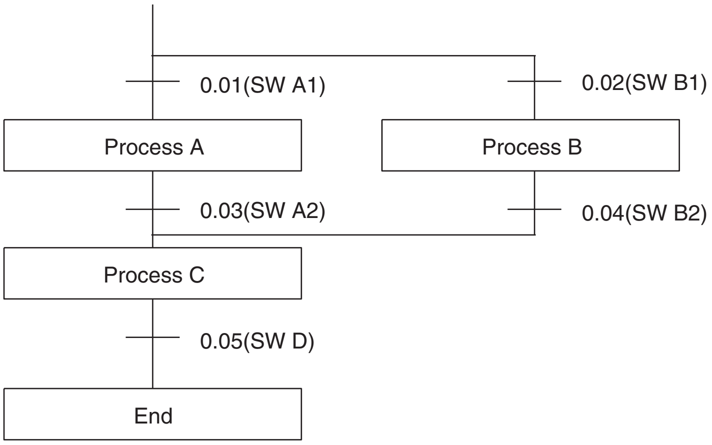
(1) SW1 ON:
    Solenoid 1 operates } Process A
    Conveyor 1 operates
(2) SW2 ON stops the previous process
    Robot hand operates } Process B
    Conveyor 2 operates
(3) SW3 ON stops the previous process
    Photo microsensor operates (for part inspection)
    Conveyor 3 operates } Process C
    Solenoid 2 operates (removal of defective items)
(4) SW4 ON stops the previous process

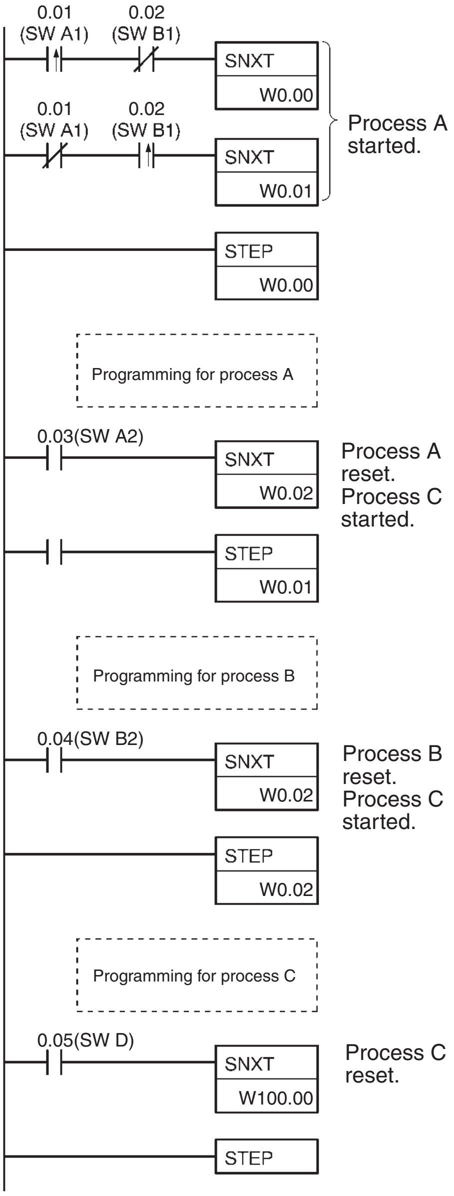
Additional Information

When another process is started from within a process (when an SNXT instruction turns ON), all outputs of the current process are turned OFF within that cycle.

(2) Branching Execution







Explanation of operation
Products are sorted by the guides by weight.

(1) SWA1 ON:
  Conveyor (A) operates } Process A
  Machine (A) operates

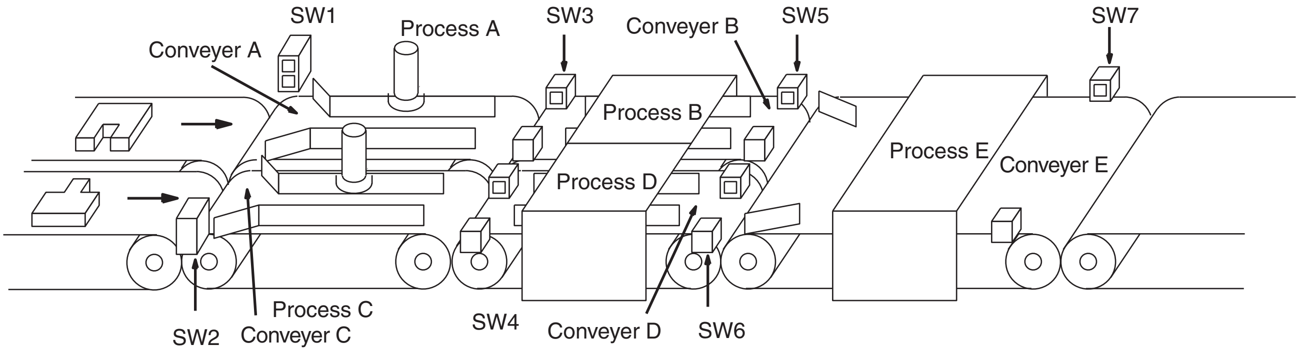
(2) SWB1 ON:
  Conveyor (B) operates } Process B
  Machine (B) operates

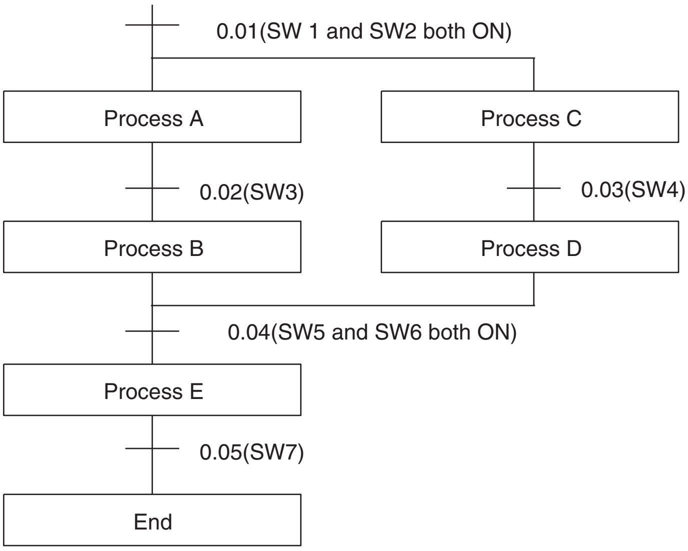
(3) SWA2 ON stops process A
  Printing machine operates (down) } Process C
  UP by SWC2 ON

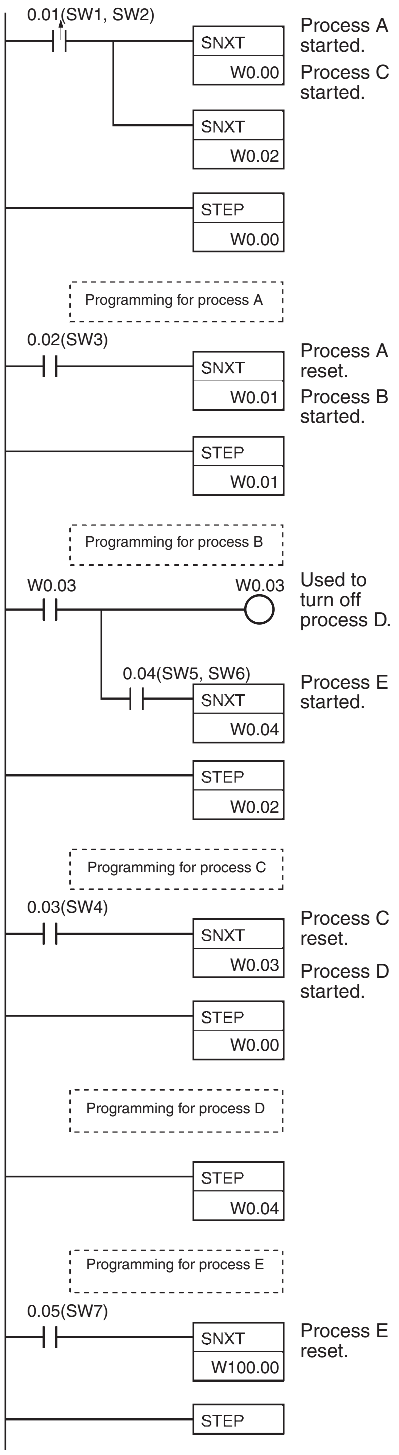
(4) SWB2 ON stops process B
  Printing machine operates (down) } Process C
  UP by SWC2 ON

(5) SWD ON stops the printing machine

(3) Parallel Execution







Explanation of operation

(1) SW1, SW2 ON:
- Conveyor (A) operates
- Work (A) operates

Process A

- Conveyor (C) operates
- Work (C) operates

Process C

(2) SW3 ON:
- Process (A) stops
- Conveyor (B) operates
- Work (B) operates

Process B

(3) SW4 ON:
- Process (C) stops
- Conveyor (D) operates
- Work (D) operates

Process D

(4) SW5, SW6 ON:
- Process (B) stops
- Process (D) stops
- Conveyor (E) operates
- Work (E) operates

Process E

(5) SW7 ON:
- Process (E) stops

Note: When processes (B) and (D) are in operation and SW5 and SW6 turn ON, it is judged that processes (B) and (D) are finished. Execution of SNXT W0.04 turns OFF the outputs of process (B) and W0.03 turns OFF.
STEP W0.03 judges ON to OFF and turns OFF the outputs of process (D).

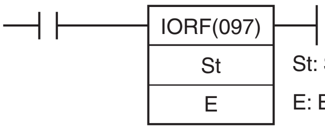
Additional Information

The STEP instruction turns all outputs in its process OFF when ON changes to OFF.

Basic I/O Unit Instructions

IORF

Instruction	Mnemonic	Variations	Function code	Function
I/O REFRESH	IORF	@IORF	097	Refreshes the specified I/O words.

	IORF
Symbol	 St: Starting word E: End word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
St	Starting word	---	Variable
E	End word	---	Variable

St: Starting Word

CIO 0000 to CIO 0999 (I/O Area) or
CIO 2000 to CIO 2959 (Special I/O Unit Area)

E: End Word

CIO 0000 to CIO 0999 (I/O Area) or
CIO 2000 to CIO 2959 (Special I/O Unit Area)

Note St and E must be in the same memory area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
St, E	OK	---	---	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	• ON if St is greater than E. • ON if St and E are in different memory areas. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.

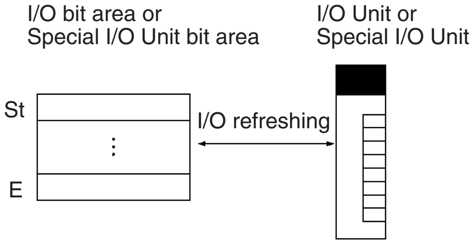
Units Refreshed by IORF(097)

Unit position	On the CPU rack or an expansion rack (excluding SYSMAC BUS Slave Racks)		Refreshable by FIORF(255)
Unit Type	Basic I/O Units	CS series Basic I/O Units	Yes
		C200H Basic I/O Units (CS Series only)	Yes
		C200H Group-2 High-density Basic I/O Units (CS Series only)	Yes
		CJ-Series Basic I/O Units	Yes
	Special I/O units (CIO words of all special I/O units, including high-density I/O units)		Yes

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 (!).

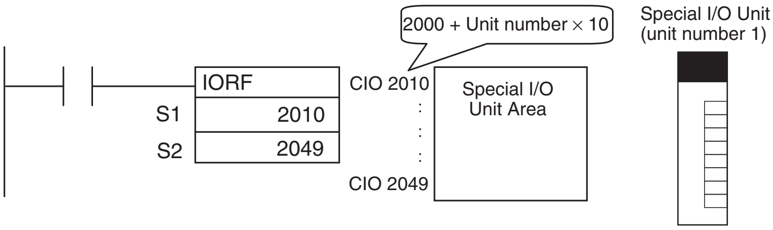
Function

IORF(097) refreshes the I/O words between St and E, inclusively. IORF(097) is used to refresh words allocated to Basic I/O Units or Special I/O Units mounted on the CPU Rack or Expansion Racks. Basic I/O Units are allocated words between CIO 0000 and CIO 0999, and Special I/O Units are allocated words between CIO 2000 and CIO 2959.



When refreshing words for a Special I/O Unit, always include the first word allocated to the Unit in the range of words being refreshed. All words allocated to the Special I/O Unit will be refreshed even if they are not included in the refresh range.

For example, a High-speed Counter Unit that is set to unit number 1 and that uses four unit numbers would be allocated the 40 words from CIO 2010 to CIO 2049. To refresh words allocated to this Unit, CIO 2010 must be specified in IORF(097). All words from CIO 2010 to CIO 2049 will be refreshed.



Hint

The following table shows how IORF(097) differs from FIORF(225) and DLNK(226).

Instruction	Operation
IORF(097)	• I/O refreshing of words used by Basic I/O Units • I/O refreshing of the CIO words and DM words used by Special I/O Units
FIORF(225)	• I/O refreshing of the CIO words and DM words used by a Special I/O Unit
DLNK(226)	• I/O refreshing of the CS1 CPU Bus Unit Area in the CIO Area (25 words) • I/O refreshing of the CS1 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

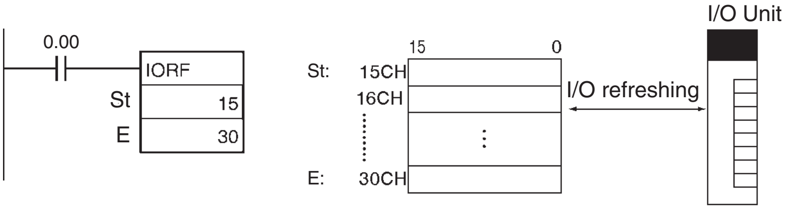
Precaution

- IORF(097) can be used in an interrupt task, which allows high-speed processing of specific I/O data with an interrupt. If IORF(097) is used in an interrupt task, always disable cyclic refreshing of the specified Special I/O Unit by turning ON the corresponding Special I/O Unit Cyclic Refreshing Disable Bit in the PLC Setup.
- When cyclic refreshing of the specified Special I/O Unit is enabled in the PLC Setup (the corresponding Special I/O Unit Cyclic Refreshing Disable Bit is OFF), a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A40213) will go ON in the following cases.
  - Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225).
  - Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).
- When cyclic refreshing of a Special I/O Unit is disabled, execute IORF(097) or FIORF(225) to refresh the Unit's data within 11 seconds after program execution starts. If IORF(097) or FIORF(225) is not executed within 11 seconds to refresh the Unit's data, a CPU Unit Monitor Error will occur in the Special I/O Unit and the ERH and RUN Indicators will be lit.
- 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.
- An error will occur if words in both the I/O Area (CIO 0000 to CIO 0999) and the Special I/O Unit Area (CIO 2000 to CIO 2959) are specified for the same instruction.
- Both C200H Special I/O Units and CS Special I/O Units can be refreshed using the same instruction. (CS Series only)
- All of the words allocated to C200H Group-2 High-density I/O Units must be refreshed at one time. The Unit's I/O words will be refreshed if the first word allocated to the Unit is in the specified range of I/O words. (The Unit's words will not be refreshed if the starting word is after the first word allocated to the Unit, but they will be refreshed even if the end word is before the last word allocated to the Unit.) (CS Series only)
- I/O refreshing will not be performed for Units for which an I/O table error has occurred. (CS Series only)
- The I/O refreshing initiated by IORF(097) will be stopped midway if an I/O bus error occurs during I/O refreshing.

Example Programming

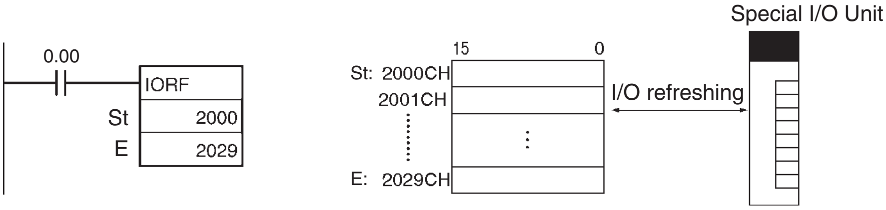
Refreshing Words in the I/O Area

The following example shows how to refresh 16 words from CIO 15 to CIO 30 when CIO 0.00 turns ON.



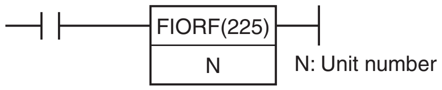
Refreshing Words in the Special I/O Unit Area

The following example shows how to refresh 30 words from CIO 2000 to CIO 2029 when CIO 0.00 turns ON.



FIORF

Instruction	Mnemonic	Variations	Function code	Function
SPECIAL I/O UNIT I/O REFRESH	FIORF	@FIORF	225	Performs I/O refreshing immediately for the specified Special I/O Unit's allocated CIO Area and DM Area words with the specified unit number.

Symbol	FIORF
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Unit number	UINT	1

N: Unit number

Specifies the Special I/O Unit's unit number (0000 to 005F hex or 0 to 95 decimal).

Note If the Special I/O Unit uses more than one unit number, specify the lowest unit number.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the specified unit number is not between 0000 and 005F hex (between 0 and 95 decimal). • ON if the PLC does not have a Special I/O Unit with the unit number specified by N. • ON if the specified Special I/O Unit uses more is allocated words for two or more unit numbers, but the unit number specified by N is not the lowest of those unit numbers. • OFF in all other cases.
Equals Flag	P_EQ	• ON if the I/O refreshing was completed normally. • OFF if FIORF(225) was executed while the specified Special I/O Unit was being refreshed during cyclic refreshing.

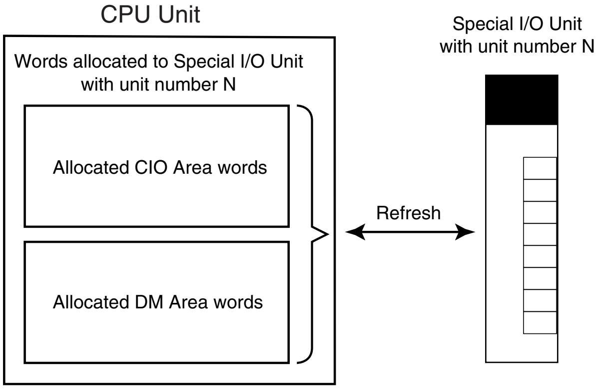
Units Refreshed by FIORF(225)

Rack	Unit type	Refreshable by FIORF(255)
CPU Rack or Expansion I/O Rack (Not applicable to Units on SYSMAC BUS Slave Racks.)	Basic I/O Units	No
	The following areas allocated to a Special I/O Unit (The words allocated to the specified Unit are refreshed together.) • Allocated CIO Area words • Allocated DM Area words	Yes
	CPU Bus Units	No

Function

FIORF(225) performs immediate I/O refreshing of the CIO Area words and DM Area words allocated to the Special I/O Unit with the unit number specified by N. Refer to the Special I/O Unit's Operation Manual for details on the data area words that are immediately refreshed.

A Special I/O Unit's regular cyclic I/O refreshing can be disabled in the PLC Setup (by turning ON the Unit's Special I/O Unit Cyclic Refresh Disable Bit), and I/O refreshing can be performed with the Unit only when necessary by executing FIORF(225). This function allows a particular Special I/O Unit's data to be refreshed when necessary, without increasing the cyclic I/O refreshing time at other times.



Hint

The following table shows how FIORF(225) differs from IORF(097) and DLNK(226).

Instruction	Operation
IORF(097)	• I/O refreshing of words used by Basic I/O Units • I/O refreshing of the CIO words and DM words used by Special I/O Units
FIORF(225)	• I/O refreshing of the CIO words and DM words used by a Special I/O Unit
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

FIORF(225) and IORF(097) both refresh the words allocated to Special I/O Units, but differ in the following ways.

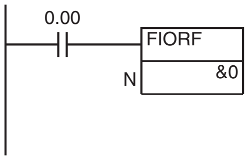
- FIORF(225) has a faster instruction execution time.
- With FIORF(225), the relevant words are specified by the unit number rather than word addresses.

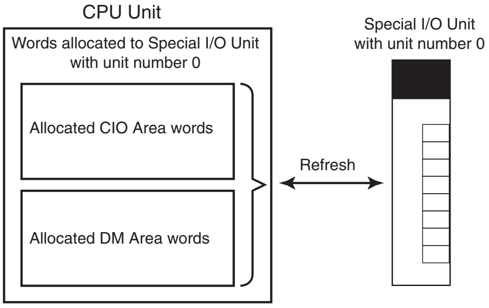
Precaution

- FIORF(225) can be used in an interrupt task, which allows high-speed processing of specific I/O data with an interrupt. If FIORF(225) is used in an interrupt task, always disable cyclic refreshing of the specified Special I/O Unit by turning ON the corresponding Special I/O Unit Cyclic Refreshing Disable Bit in the PLC Setup.
- When cyclic refreshing of the specified Special I/O Unit is enabled in the PLC Setup (the corresponding Special I/O Unit Cyclic Refreshing Disable Bit is OFF), a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A402.13) will go ON in the following cases.
  - Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225).
  - Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).
- When cyclic refreshing of a Special I/O Unit is disabled, execute IORF(097) or FIORF(225) to refresh the Unit's data within 11 seconds after program execution starts. If IORF(097) or FIORF(225) is not executed within 11 seconds to refresh the Unit's data, a CPU Unit Monitor Error will occur in the Special I/O Unit and the ERH and RUN Indicators will be lit.
- I/O refreshing by FIORF(225) will be stopped if an I/O Bus Error occurs while during I/O refreshing.

Example Programming

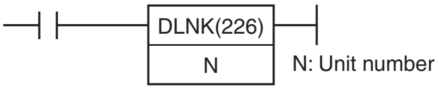
When CIO 0.00 is ON, FIORF(225) immediately refreshes the CIO Area and DM Area words allocated to the Special I/O Unit set as unit number 0.





DLNK

Instruction	Mnemonic	Variations	Function code	Function
CPU BUS UNIT I/O REFRESH	DLNK	@DLNK	226	Performs I/O refreshing immediately for the CPU Bus Unit with the specified unit number.

Symbol	DLNK
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Unit number	UINT	1

N: Unit number

Specifies the CPU Bus Unit's unit number (0000 to 000F hex or 0 to 15 decimal).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

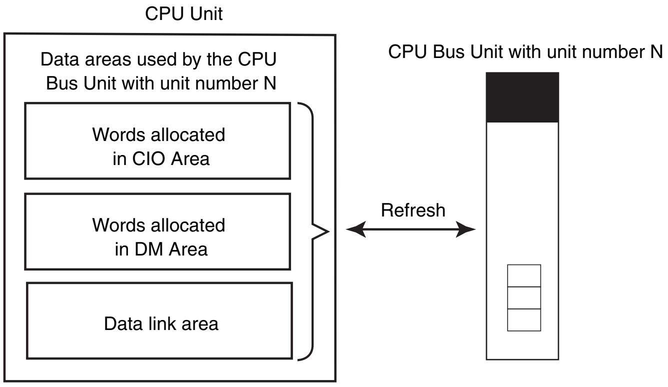
Name	Label	Operation
Error Flag	P_ER	• ON if the specified unit number is not between 0000 and 000F hex (between 0 and 15 decimal). • ON if the PLC does not have a CPU Bus Unit with the specified unit number. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.
Equals Flag	P_EQ	• OFF if the I/O refreshing could not be performed because the CPU Bus Unit was refreshing data. • OFF if there was a CPU Bus Unit Error or CPU Bus Unit Setup 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.

Function

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 *Hint* 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 or SYSMAC Link Unit	Data link refreshing
DeviceNet Unit (Does not include C200H DeviceNet Master Units.)	Remote I/O communications refreshing



Hint

The following table shows how DLNK(226) differs from FIORF(225) and IORF(097).

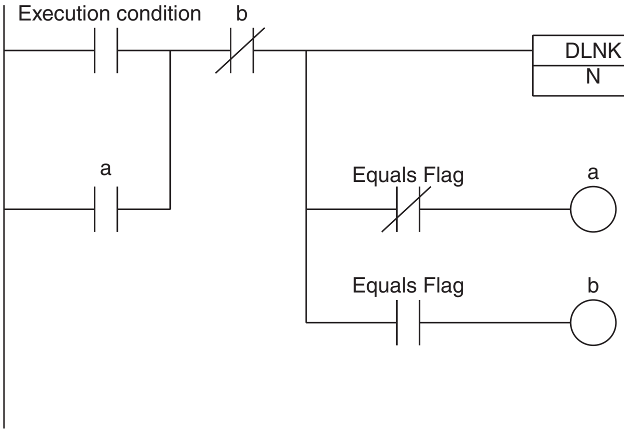
Instruction	Operation
IORF(097)	• I/O refreshing of words used by Basic I/O Units • I/O refreshing of the CIO words and DM words used by Special I/O Units
FIORF(225)	• I/O refreshing of the CIO words and DM words used by a Special I/O Unit
DLNK(226)	• I/O refreshing of the CS1 CPU Bus Unit Area in the CIO Area (25 words) • I/O refreshing of the CS1 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

DLNK(226) refreshes data between the CPU Unit and specified CPU Bus Unit. There are two special factors to consider when using DLNK(226):

- 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.
- 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.

Precaution

- 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.

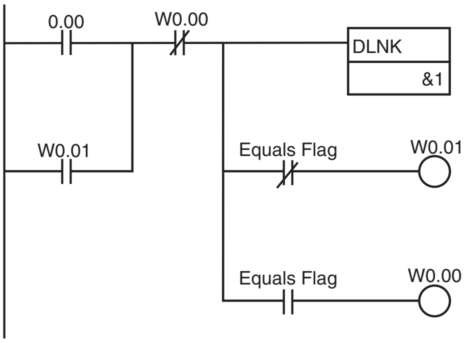


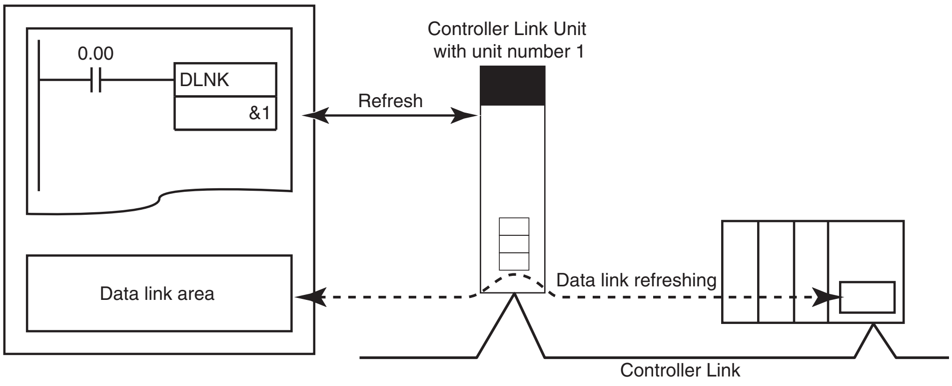
- I/O refreshing will not be performed if a CPU Bus Unit Error (A402.07) or CPU Bus Unit Setup Error (A402.03) has occurred in the specified CPU Bus Unit.
- I/O refreshing will be stopped if an I/O Bus Error occurs while I/O refreshing is being performed by DLNK(226).

Example Programming

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.



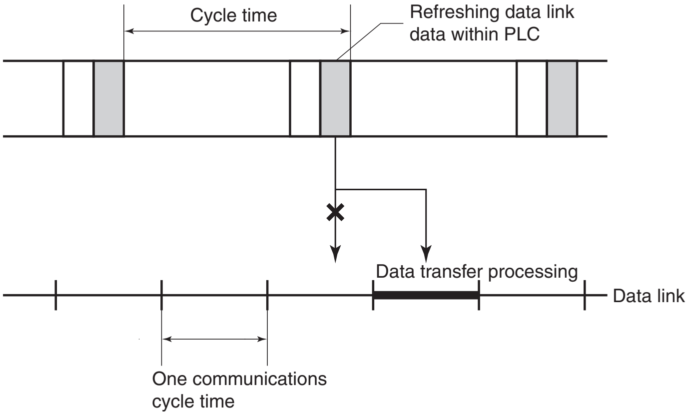


The actual timing for data link area refreshing in this example is as follows:

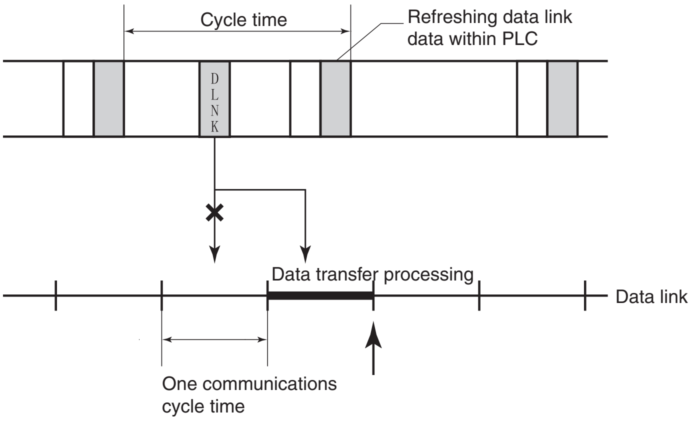
- When transmitting: Data is transmitted over the network the next time that the token right is acquired. (The transmitted data is delayed up to 1 communications cycle time max.)
- When receiving: The data that is input was received from the network the last time that the token right was acquired. (The data received is delayed up to 1 communications cycle time max.)

Examples of Data Transfer Processing:

- Transferring Data from the Previous I/O Refreshing

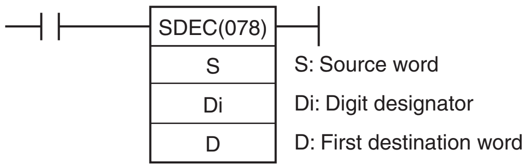


- Transferring Data with Execution of DLNK(226)



SDEC

Instruction	Mnemonic	Variations	Function code	Function
7-SEGMENT DECODER	SDEC	@SDEC	078	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.

	SDEC
Symbol	

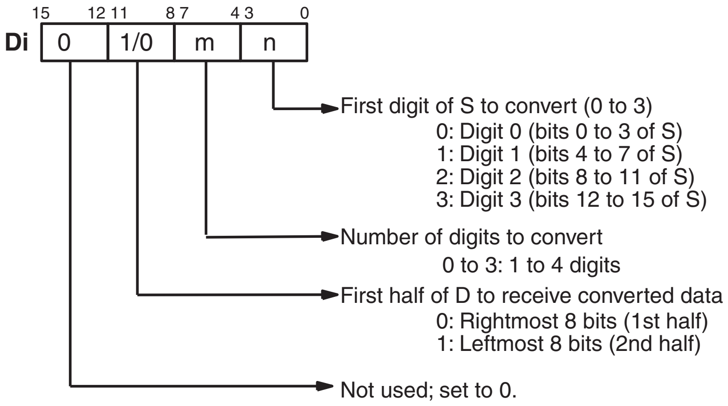
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	UINT	1
Di	Digit designator	UINT	1
D	First destination word	UINT	Variable

Di: Digit designator



First digit of S to convert (0 to 3)
0: Digit 0 (bits 0 to 3 of S)
1: Digit 1 (bits 4 to 7 of S)
2: Digit 2 (bits 8 to 11 of S)
3: Digit 3 (bits 12 to 15 of S)

Number of digits to convert
0 to 3: 1 to 4 digits

First half of D to receive converted data
0: Rightmost 8 bits (1st half)
1: Leftmost 8 bits (2nd half)

Not used; set to 0.

● Operand Specifications

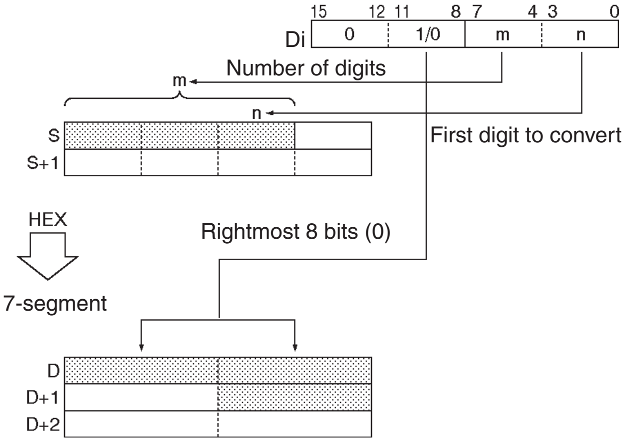
Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
Di											OK							
D											---	---						

Flags

Name	Label	Operation
Error Flag	ER	• ON if settings in Di are not within the specified ranges. • OFF in all other cases.

Function

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.

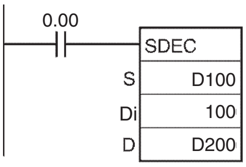


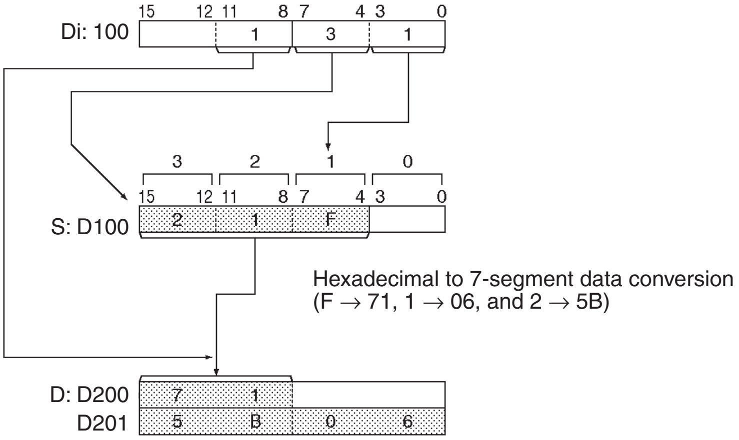
Precaution

- 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.

Example Programming

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 100.



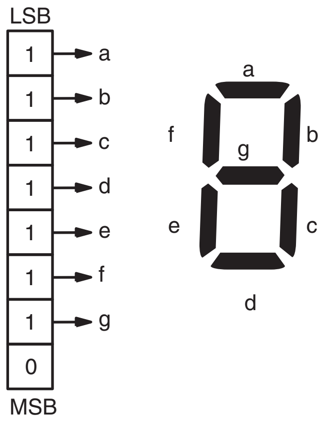


Hexadecimal to 7-segment data conversion
(F → 71, 1 → 06, and 2 → 5B)

7-segment Data

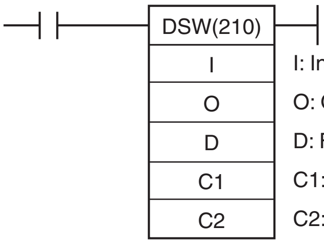
The following table shows the data conversions from a hexadecimal digit (4 bits) to 7-segment code (8 bits).

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



DSW

Instruction	Mnemonic	Variations	Function code	Function
DIGITAL SWITCH INPUT	DSW	---	210	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.

	DSW
Symbol	 I: Input word O: Output word D: First result word C1: Number of digits C2: System word

Applicable Program Areas

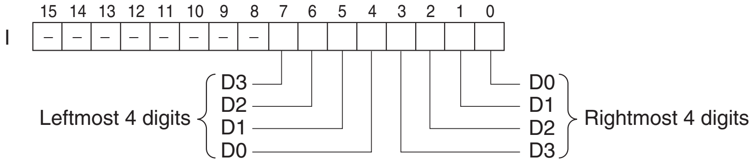
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
I	Input word	UINT	1
O	Output word	UINT	1
D	First result word	WORD	Variable
C1	Number of digit	UINT	1
C2	System word	WORD	1

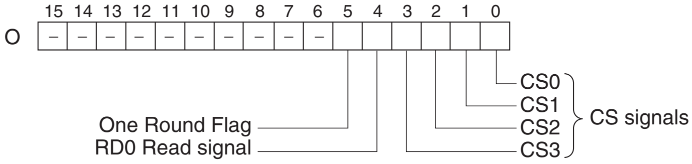
I: Input Word (Data Line D0 to D3 Inputs)

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.



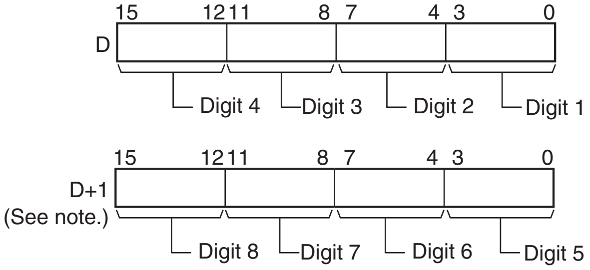
O: Output Word (CS/RD Control Signal Outputs)

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.



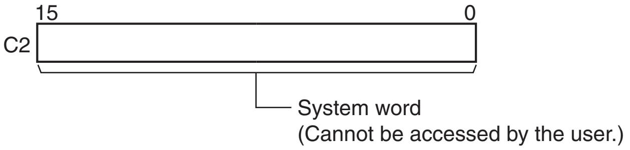
Note: Only when C1 = 0001 hex to read 8 digits.

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
I, O	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D												---						
C1	---	---	---	---	---	---	---	---	---	---	OK			---				
C2	OK	OK	OK	OK	OK	OK	OK	OK	OK		---	OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	OFF

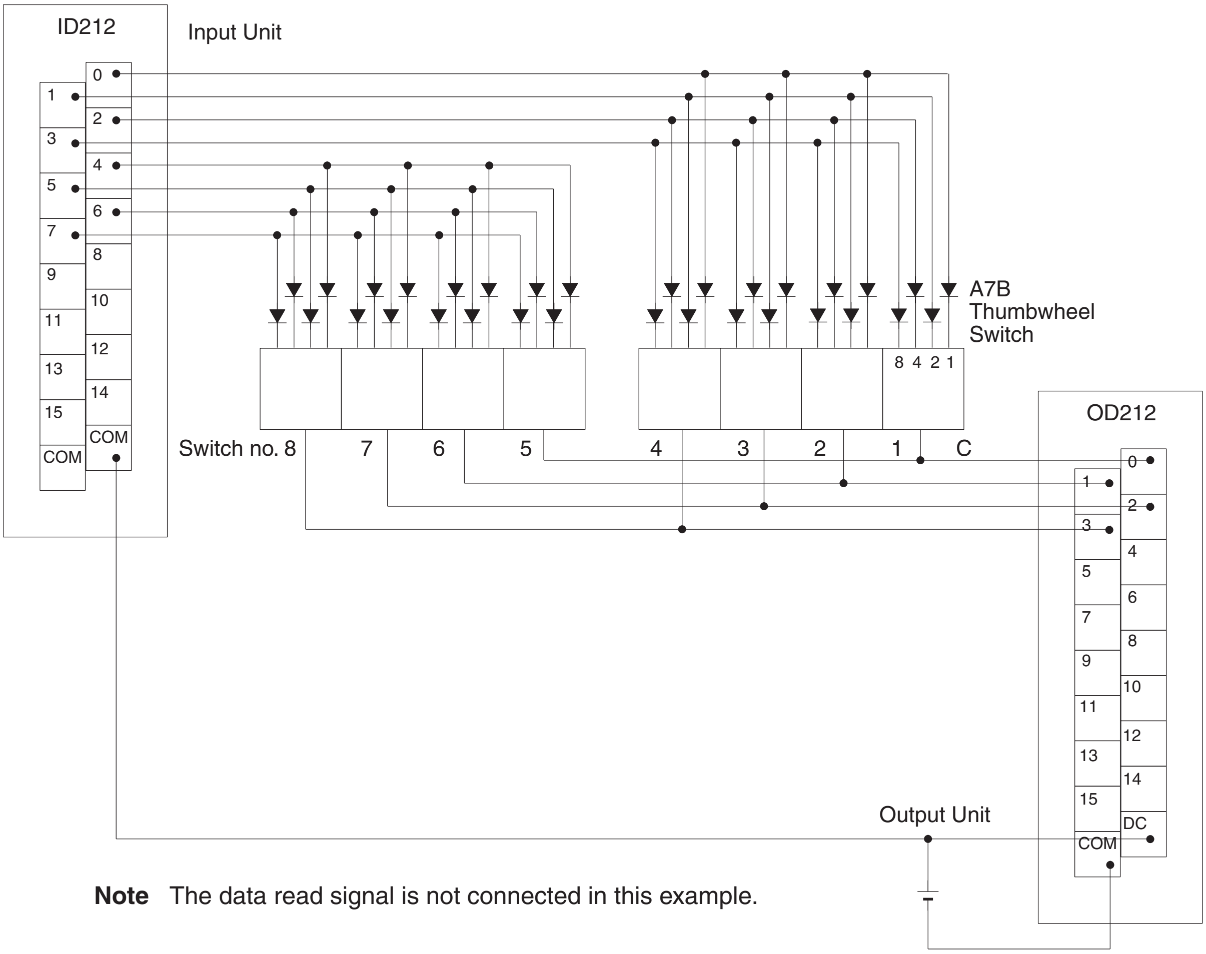
Function

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.

● 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.

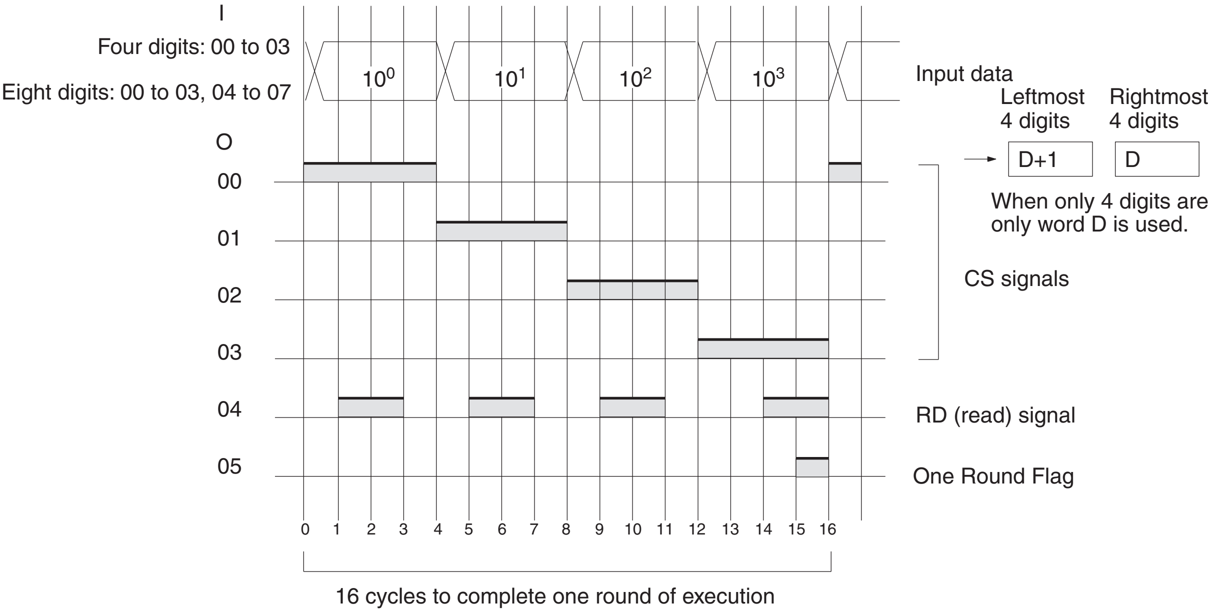


Note The data read signal is not connected in this example.

The inputs and outputs can be connected to the following kinds of Basic I/O Units and High-density I/O Units as long as they are not mounted in a SYSMAC BUS Remote I/O Rack.

- DC Input Units with 8 or more input points
- Transistor Output Units with 8 or more output points

● Timing Chart



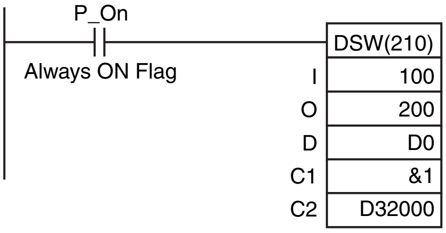
Precaution

- 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.
- I/O refreshing must be performed with the I/O Unit connected to the digital switch or thumbwheel switch after each DSW(210) instruction is executed. If the cycle time is short, operation may not be correct depending on the ON/OFF response time of the I/O Unit. If necessary, use an I/O Unit with a short ON/OFF response time or use the minimum cycle time function to adjust the cycle time. Do not connect the digital switch or thumbwheel switch to the following Units.
  - Basic I/O Units or High-density I/O Units mounted in a SYSMAC BUS Remote I/O Slave Rack
  - Communications Slaves (DeviceNet or CompoBus/S Slaves)
- 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 (unlike the C200HX/HG/HE and CQM1H Series).

Example Programming

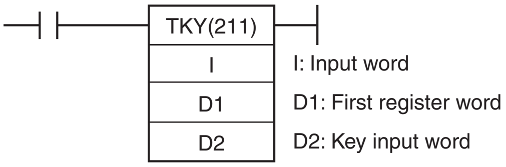
In this example, DSW(210) is used to read an 8-digit number from a digital switch and outputs the resulting value constantly to D0 and D1. The digital switch is connected through CIO 100 (allocated to a CS1W-ID211 16-point DC Input Unit) and CIO 200 (allocated to a CS1W-OD211 16-point Transistor Output Unit).

D32000 is used as the system word.



TKY

Instruction	Mnemonic	Variations	Function code	Function
TEN KEY INPUT	TKY	@TKY	211	Reads numeric data from a ten-key keypad connected to an Input Unit and stores up to 8 digits of BCD data in the specified words.

	TKY
Symbol	

Applicable Program Areas

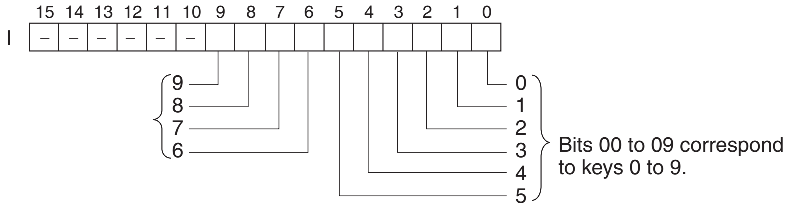
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
I	Input word	UINT	1
D1	First register word	UDINT	2
D2	Key input word	UINT	1

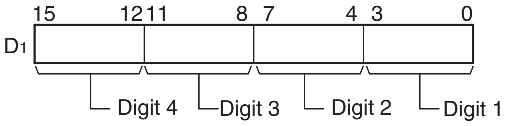
I: Input Word (Data Line Inputs)

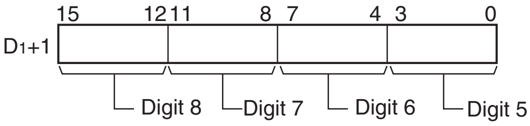
Specify the input word allocated to the Input Unit and connect the ten-key keypad's 0 to 9 data lines to the Input Unit as shown in the following diagram.



D1: First Register Word

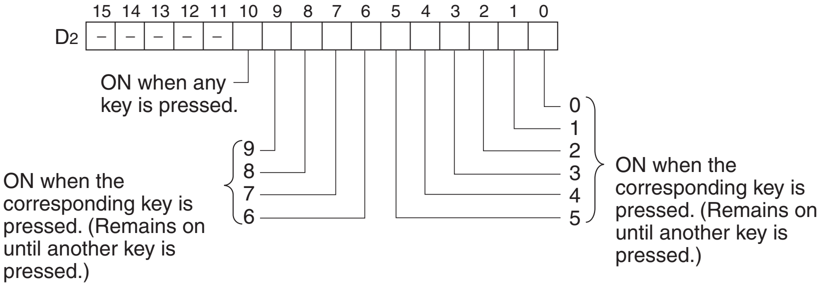
Specifies the leading word address where the ten-key keypad's numeric input (up to 8 digits) will be stored.





D2: Key Input Word

Bits 00 to 10 of D2 indicate key inputs. When one of the keys on the keypad (0 to 9) has been pressed, the corresponding bit of D2 (0 to 9) is turned ON. Bit 10 of D2 will be ON while any key is being pressed.



Note TKY(211) does not require a system word, unlike other I/O instructions such as HKY(212).

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
I	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D1												---						
D2												OK						

Flags

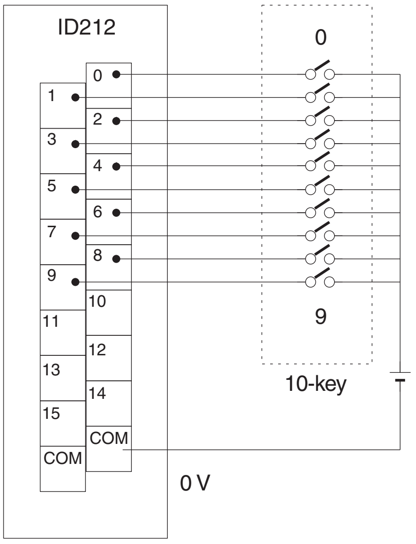
Name	Label	Operation
Error Flag	P_ER	OFF

Function

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 D1 and D1+1. In addition, each time that a key is pressed, the corresponding bit in D2 (0 to 9) will be turned ON and remains ON until another key is pressed. Bit 10 of D2 will be ON while any key is being pressed and OFF when no key is being pressed.

External Connections

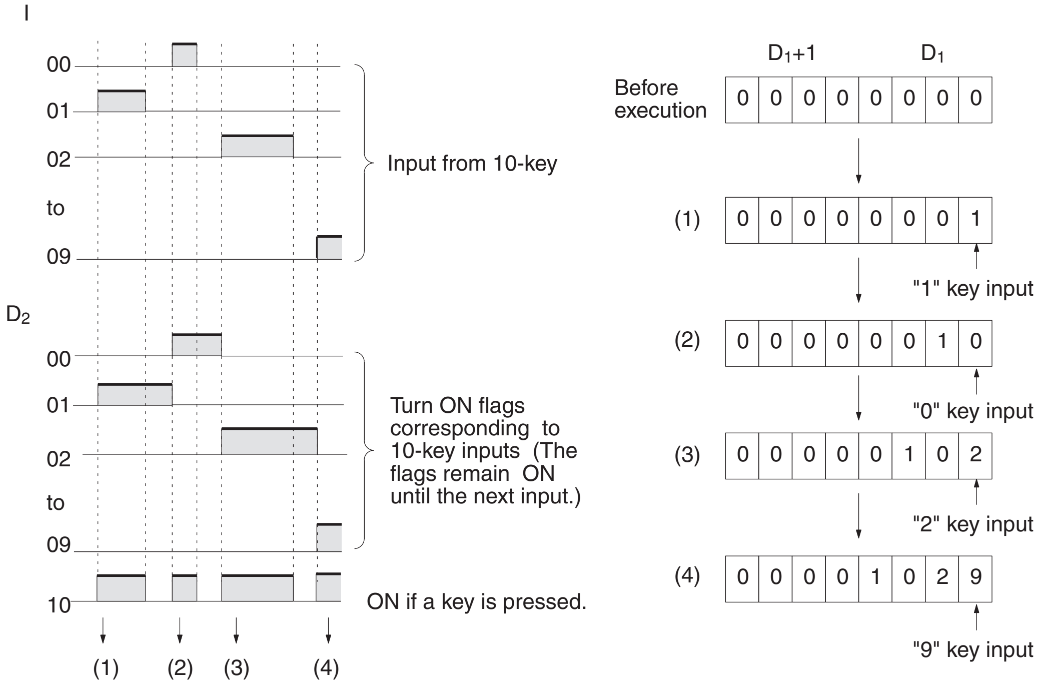
Connect the ten-key keypad so that the switches for keys 0 through 9 are input to contacts 0 through 9 of the Input Unit, as shown in the following diagram.



DC Input Unit

The Input Unit must be a DC Input Unit or High-density Input Unit with at least 16 inputs and the Input Unit cannot be mounted in a SYSMAC BUS Remote I/O Rack.

● Timing Chart

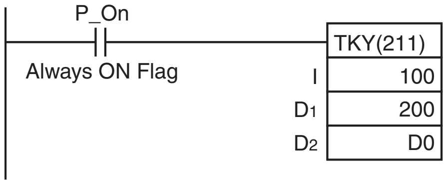


Precaution

- I/O refreshing must be performed with the I/O Unit connected to the ten-key pad after each TKY(211) instruction is executed. If the cycle time is short, operation may not be correct depending on the ON/OFF response time of the I/O Unit. If necessary, use an I/O Unit with a short ON/OFF response time or use the minimum cycle time function to adjust the cycle time. Do not connect the ten-key keypad to the following Units.
    - Basic I/O Units or High-density I/O Units mounted in a SYSMAC BUS Remote I/O Slave Rack
    - Communications Slaves (DeviceNet or CompoBus/S Slaves)
- The two-word register in D1 and D1+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 D1. The other digits of D1, D1+1 are shifted left and the most significant digit of D1+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 (unlike the C200HX/HG/HE and CQM1H Series).

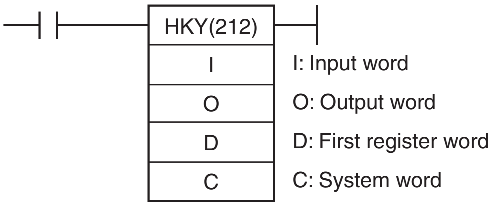
Example Programming

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 100 (allocated to a CS1W-ID211 16-point DC Input Unit).



HKY

Instruction	Mnemonic	Variations	Function code	Function
HEXADECIMAL KEY INPUT	HKY	---	212	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.

	HKY
Symbol	

Applicable Program Areas

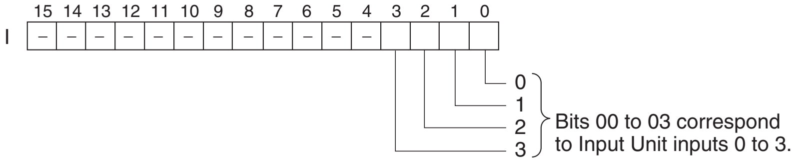
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
I	Input word	UINT	1
O	Output word	UINT	1
D	First register word	WORD	3
C	System word	WORD	1

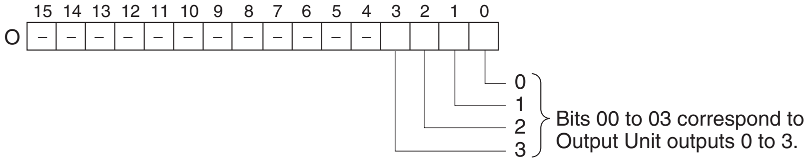
I: Input Word (Data Line D0 to D3 Inputs)

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.



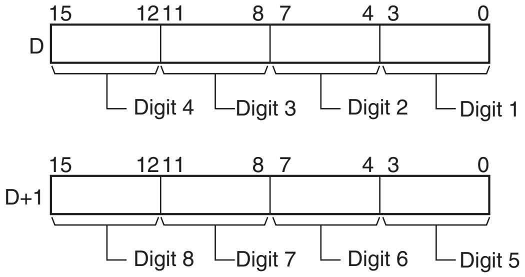
O: Output Word (Selection Signal Outputs)

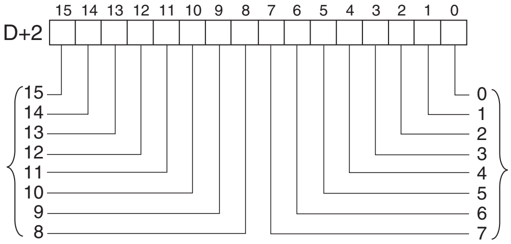
Specify the output word allocated to the Output Unit and connect the hexadecimal keypad's selection signals to the Output Unit as shown in the following diagram.



D: First Register Word

Specifies the leading word address where the hexadecimal keypad's numeric input (up to 8 digits) will be stored.

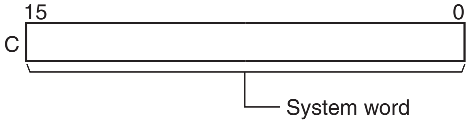




ON when the corresponding key is pressed. (Remains on until another key is pressed.)

C: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



System word
(Cannot be accessed by the user.)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
I, O	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D												---						
C												OK						

Flags

Name	Label	Operation
Error Flag	P_ER	OFF

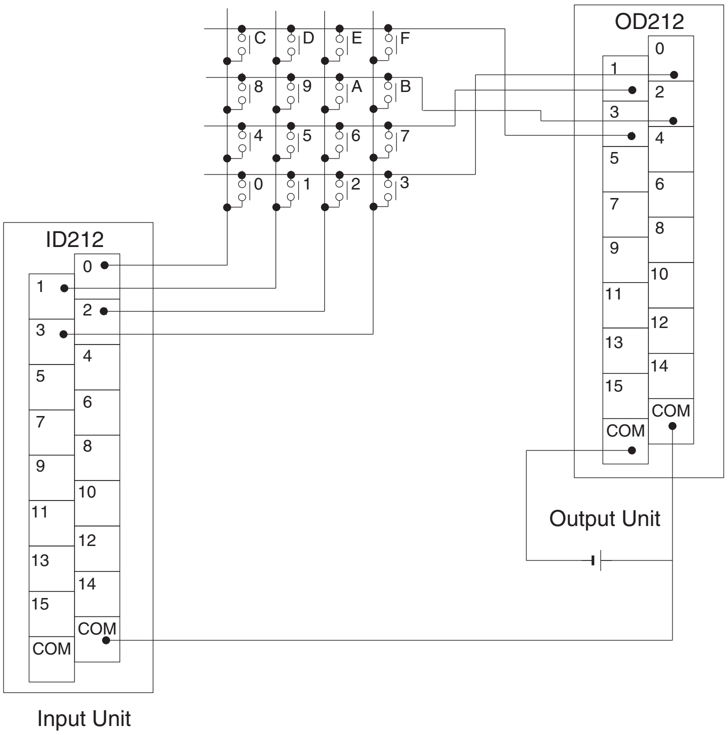
Function

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.

External Connections

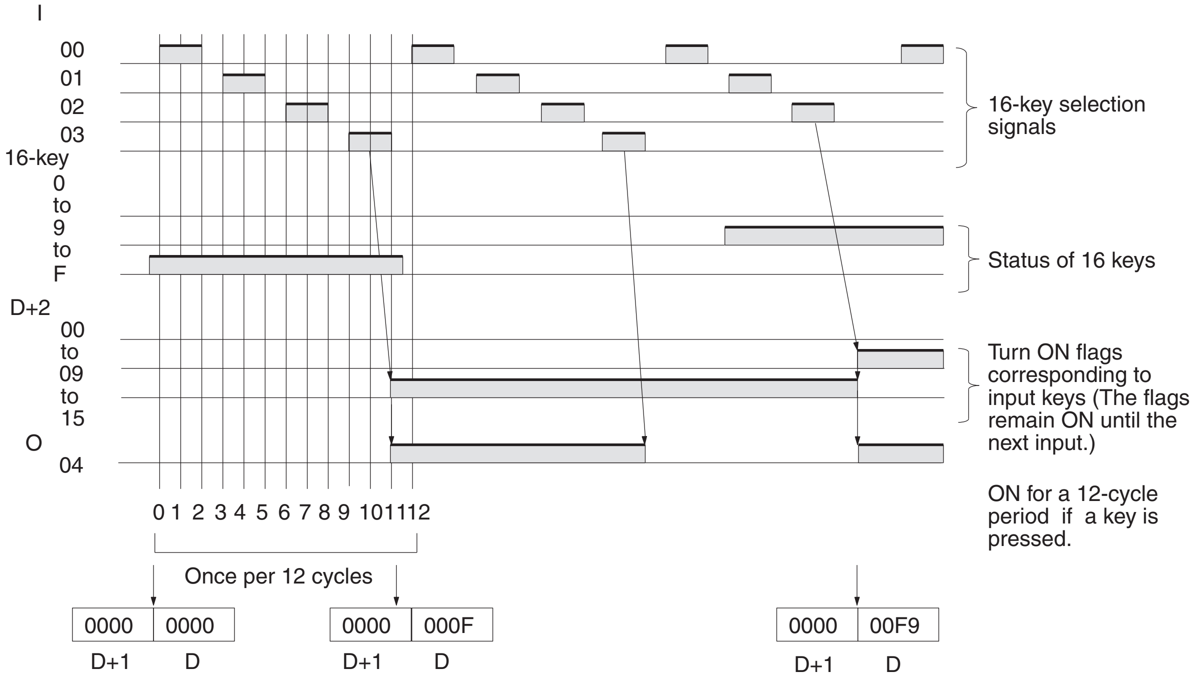
Connect the hexadecimal keypad to Input Unit contacts 0 to 3 and Output Unit contacts 0 to 3, as shown in the following diagram.



The inputs and outputs can be connected to the following kinds of Basic I/O Units and High-density I/O Units as long as they are not mounted in a SYSMAC BUS Remote I/O Rack.

- DC Input Units with 8 or more input points
- Transistor Output Units with 8 or more output points

Timing Chart

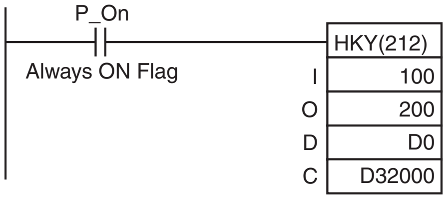


Precaution

- 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.
- I/O refreshing must be performed with the I/O Unit connected to the hexadecimal keypad after each HKY(212) instruction is executed. If the cycle time is short, operation may not be correct depending on the ON/OFF response time of the I/O Unit. If necessary, use an I/O Unit with a short ON/OFF response time or use the minimum cycle time function to adjust the cycle time. Do not connect the hexadecimal keypad to the following Units.
    - Basic I/O Units or High-density I/O Units mounted in a SYSMAC BUS Remote I/O Slave Rack
    - Communications Slaves (DeviceNet or CompoBus/S Slaves)
- 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 D1 and D1+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 D1. The other digits of D1, D1+1 are shifted left and the most significant digit of D1+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 (unlike the CQM1H Series).

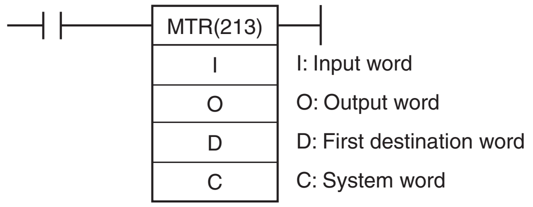
Example Programming

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 100 (allocated to a CS1W-ID211 16-point DC Input Unit) and CIO 200 (allocated to a CS1W-OD211 16-point Transistor Output Unit). D32000 is used as the system word.



MTR

Instruction	Mnemonic	Variations	Function code	Function
MATRIX INPUT	MTR	---	213	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.

	MTR
Symbol	

Applicable Program Areas

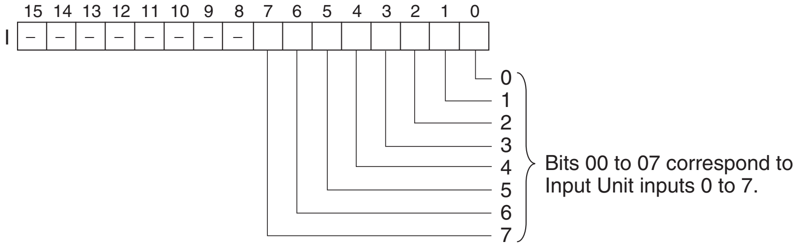
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
I	Input word	UINT	1
O	Output word	UINT	1
D	First destination word	ULINT	4
C	System word	WORD	1

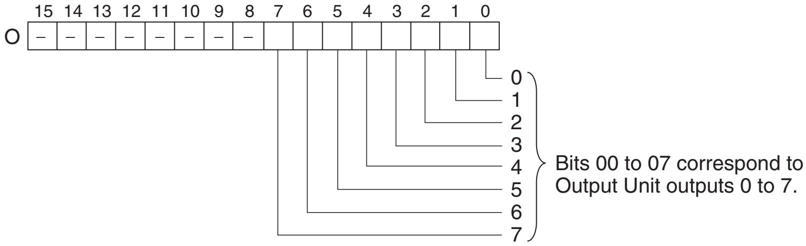
I: Input Word

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.



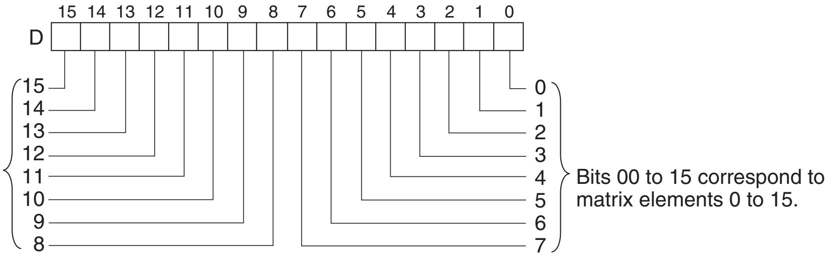
O: Output Word (Selection Signal Outputs)

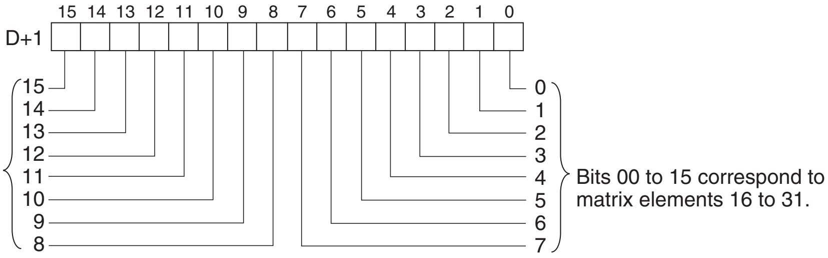
Specify the output word allocated to the Output Unit and connect the 8 selection signals to the Output Unit as shown in the following diagram.

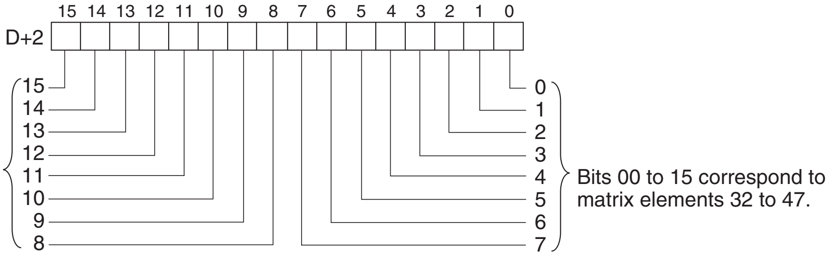


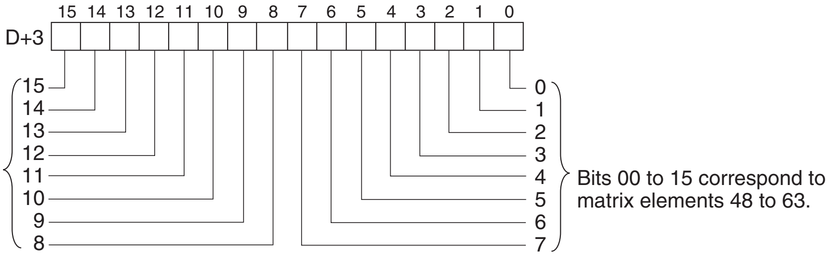
D: First Register Word

Specifies the leading word address of the 4 words that contain the data from the 8 × 8 matrix.



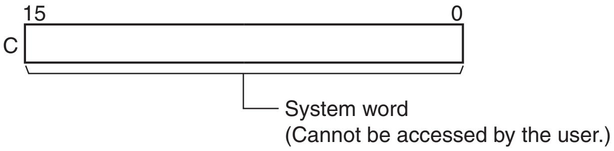






C: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
I, O	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D												---						
C												OK						

Flags

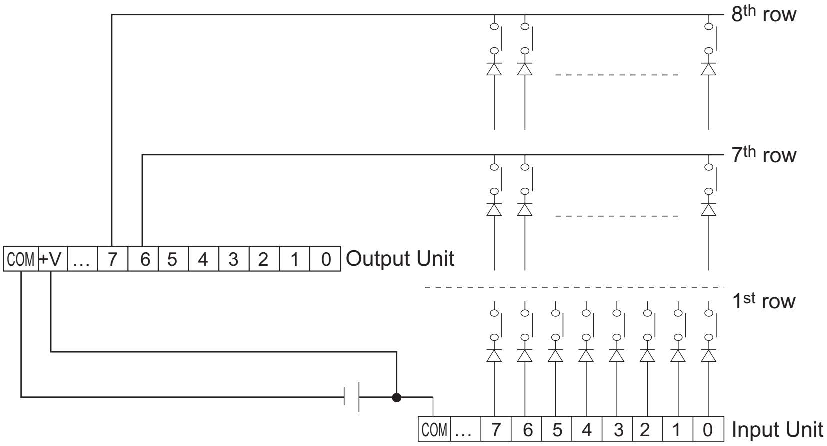
Name	Label	Operation
Error Flag	P_ER	OFF

Function

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.

● External Connections

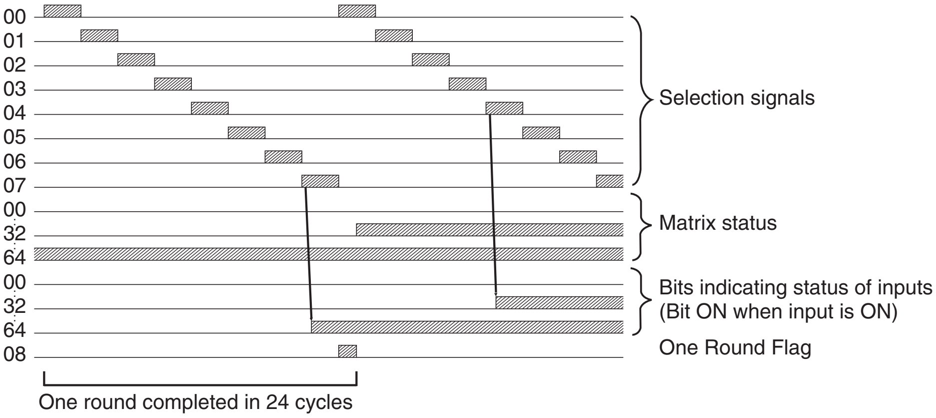
Connect the hexadecimal keypad to Input Unit contacts 0 to 7 and Output Unit contacts 0 to 7, as shown in the following diagram.



The inputs and outputs can be connected to the following kinds of Basic I/O Units and High-density I/O Units as long as they are not mounted in a SYSMAC BUS Remote I/O Rack.

- DC Input Units with 8 or more input points
- Transistor Output Units with 8 or more output points

● Timing Chart

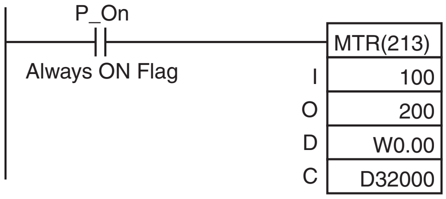


Precaution

- 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.
- I/O refreshing must be performed with the I/O Unit connected to the external matrix after each MTR(213) instruction is executed. If the cycle time is short, operation may not be correct depending on the ON/OFF response time of the I/O Unit. If necessary, use an I/O Unit with a short ON/OFF response time or use the minimum cycle time function to adjust the cycle time. Do not connect the external matrix to the following Units.
  - Basic I/O Units or High-density I/O Units mounted in a SYSMAC BUS Remote I/O Slave Rack
  - Communications Slaves (DeviceNet or CompoBus/S Slaves)
- 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 (unlike the C200HX/HG/HE and CQM1H Series).

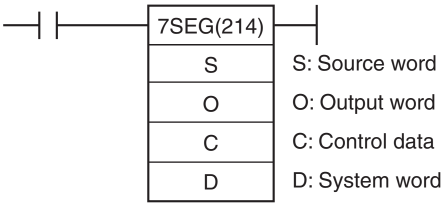
Example Programming

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 100 (allocated to a CS1W-ID211 16-point DC Input Unit) and CIO 200 (allocated to a CS1W-OD211 16-point Transistor Output Unit). D32000 is used as the system word.



7SEG

Instruction	Mnemonic	Variations	Function code	Function
7-SEGMENT DISPLAY OUTPUT	7SEG	---	214	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.

	7SEG
Symbol	

Applicable Program Areas

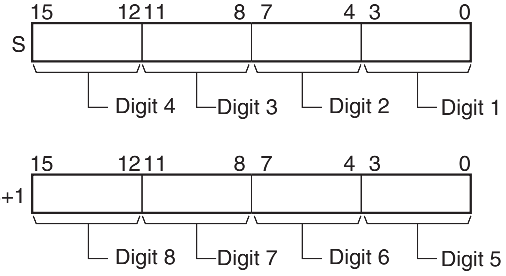
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Not allowed	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	Variable
O	Output word	UINT	1
C	Control word	#+10 decimal only	1
D	System word	WORD	1

S: Source Word

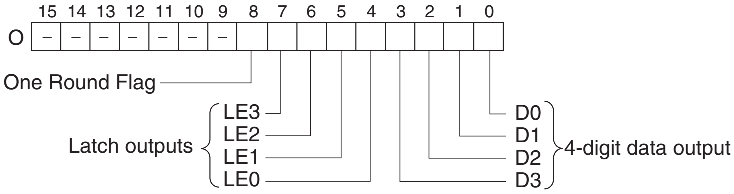
Specify the first source word containing the data that will be converted to 7-segment display data.



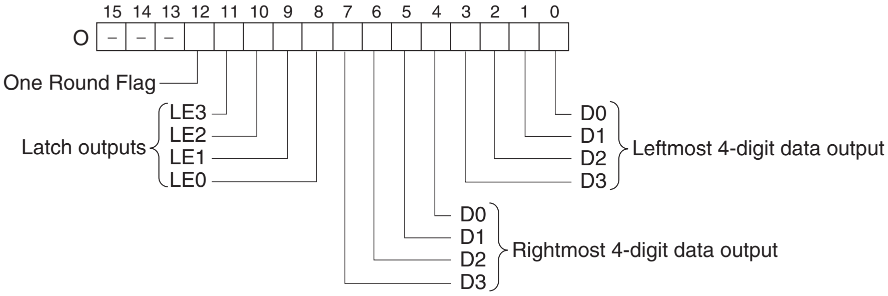
O: Output Word (Data and Latch Outputs)

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



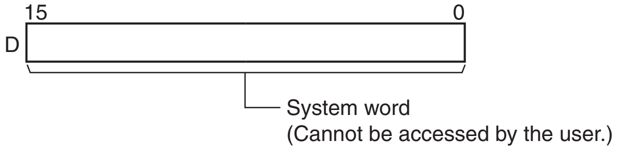
C: Control Data

The value of C (0000 to 0007 hex) 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	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
O												OK						
C	---	---	---	---	---	---	---	---	---	---	OK	---		---				
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	OFF

Function

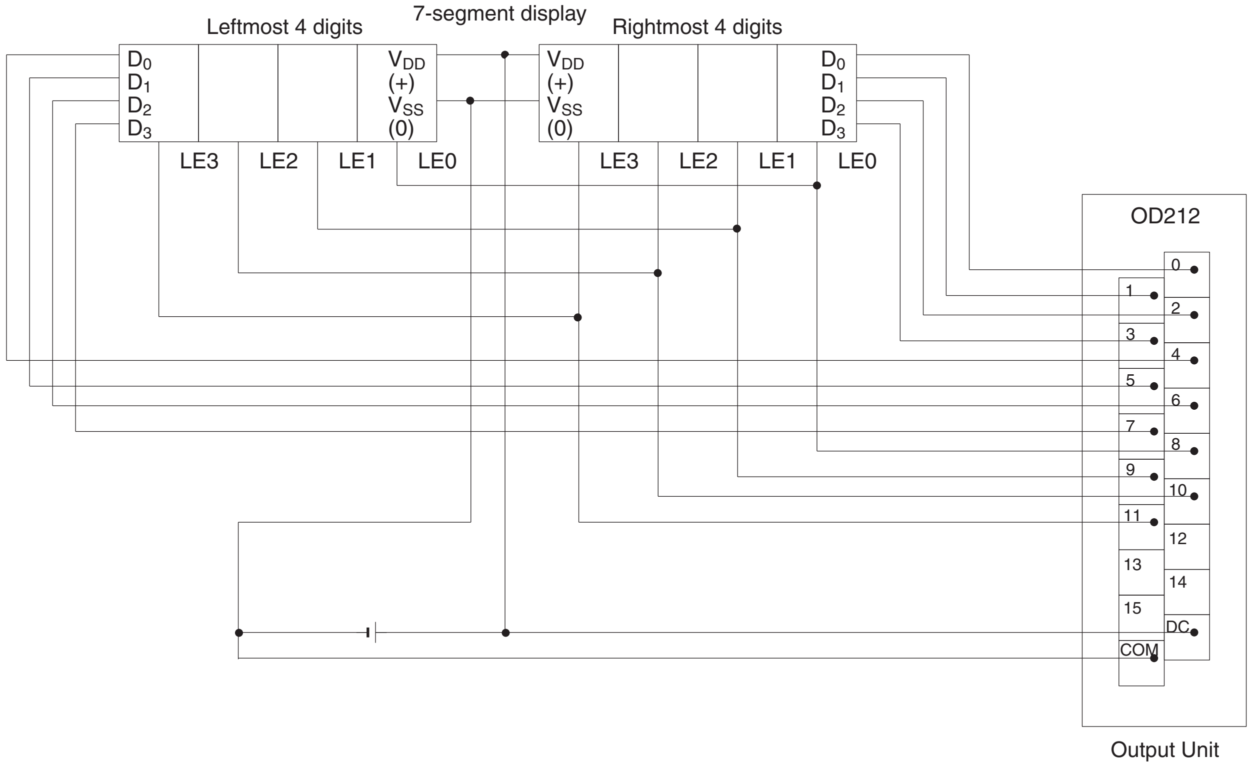
7SEG(214) reads the source data, converts it to 7-segment display data, and outputs that data (as leftmost 4 digits D0 to D3, rightmost 4 digits D0 to D3, latch output signals LE0 to LE3) to the 7-segment display connected to the output indicated by O. The value of C indicates the number of digits of 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.

● External Connections

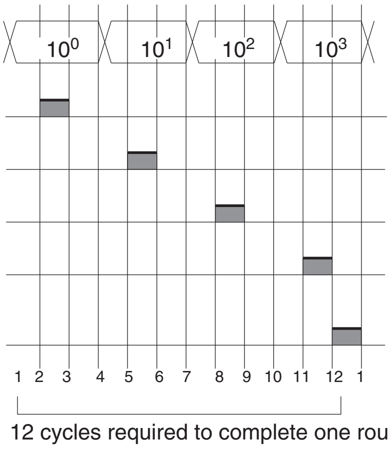
Connect the 7-segment display to the Output Unit as shown in the following diagram. This example shows an 8-digit display. With a 4-digit display, the data outputs (D0 to D3) would be connected to outputs 0 to 3 and the latch outputs (LE0 to LE3) would be connected to outputs 4 to 7. Output point 12 (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.



The inputs and outputs can be connected to the following kinds of Basic I/O Units and High-density I/O Units as long as they are not mounted in a SYSMAC BUS Remote I/O Rack.

- 4-digit display: Transistor Output Units with 8 or more output points
- 8-digit display: Transistor Output Units with 16 or more output points

● 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	 **Note** 0 to 3: Data output for word S 4 to 7: Data output for word S+1
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	

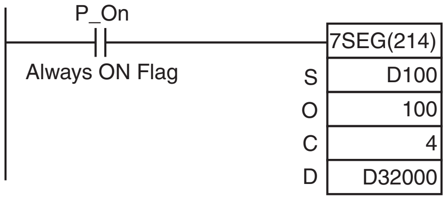
Precaution

- 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.
- I/O refreshing must be performed with the Output Unit connected to the 7-segment display after each 7SEG(214) instruction is executed. If the cycle time is short, operation may not be correct depending on the ON/OFF response time of the I/O Unit. If necessary, use an I/O Unit with a short ON/OFF response time or use the minimum cycle time function to adjust the cycle time. Do not connect the 7-segment display to the following Units.
  - Basic I/O Units or High-density I/O Units mounted in a SYSMAC BUS Remote I/O Slave Rack
  - Communications Slaves (DeviceNet or CompoBus/S Slaves)
- 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.

Example Programming

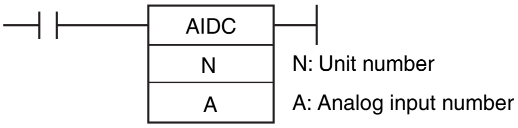
In this example, 7SEG(214) converts the 8 digits of BCD data in D100 and D101 and outputs the data through CIO 100 to a 7-segment display connected to a CS1W-OD211 16-point Transistor Output Unit.

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 4. D32000 is used as the system word, D.



AIDC

Instruction	Mnemonic	Variations	Function code	Function
ANALOG INPUT DIRECT CONVERSION	AIDC	@AIDC	216	Reads the input conversion value of the specified analog input number from the CJ1W-AD042 Analog Input Unit in Direct Conversion Mode.

	AIDC
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Unit number	UINT	1
A	Analog input number	UINT	1

N: Unit number

Specify the unit number between 0 and 95 using &0 to &95 decimal or #0000 to #005F hex.

A: Analog input number

Specify the analog input or inputs using a constant between &0 and &4 decimal or #0000 and #0004 hex.

&0 (#0000): Analog input 1 to the last analog input

&1 (#0001): Analog input 1

&2 (#0002): Analog input 2

&3 (#0003): Analog input 3

&4 (#0004): Analog input 4

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
A	---	---	---	---	---	---	---	---	---	---	OK	---		---				

Flags

Name	Label	Operation
Error Flag	ER	• ON if the unit number in N is not between &0 and &95 decimal (#0000 and #005F hex). • ON if the specified unit is not a CJ1W-AD042 Analog Input Unit. • ON if the specified CJ1W-AD042 Analog Input Unit is being initialized or restarted. • ON if there is no Unit with the unit number specified in N. • ON if AIDC(216) is executed in an interrupt task when it is also being executed in a cyclic task. • ON if an analog input that is not being used is specified in A. • ON if the specified CJ1W-AD042 Analog Input Unit is not in Direct Conversion Mode. • ON if there is a DM parameter setting error for specified CJ1W-AD042 Analog Input Unit. (The ERC indicator will be lit.) • ON if there is an analog converter error. • OFF in all other cases.
Equals Flag	=	• ON if reading the data was completed normally. • ON if reading the data was not completed normally.

Related Auxiliary Area Words and Bits

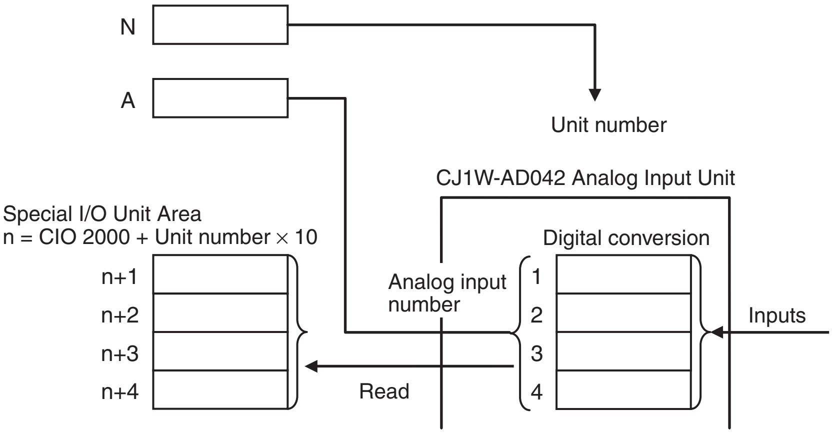
Name	Addresses	Operation
Special I/O Unit Initialization Flags	A330.00 to A335.15	ON when a Special I/O Unit is being initialized. OFF: Not being initialized. ON: Being initialized.

Function

AIDC(216) executes digital conversion for the analog input or inputs specified in A for the CJ1W-AD042 Analog Input Unit with the unit number specified in N, reads the conversion results immediately, and stores them in the Special I/O Unit Area in the CIO Area.

If 0 is specified for A, four words of data are stored. If 1 to 4 is specified for A, one word of data is stored. If 0 is specified for A, 0 will be stored for unused analog inputs.

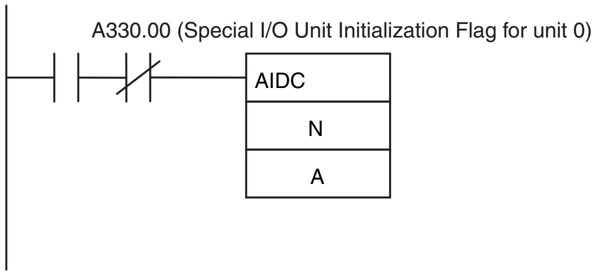
The specified CJ1W-AD042 Analog Input Unit must be in Direct Conversion Mode before AIDC(216) is executed. Refer to the *SYSMAC CS/CJ-series Analog I/O Units Operation Manual* (Cat. No. W345) for details on the Direct Conversion Mode and the number of analog inputs.



Precaution

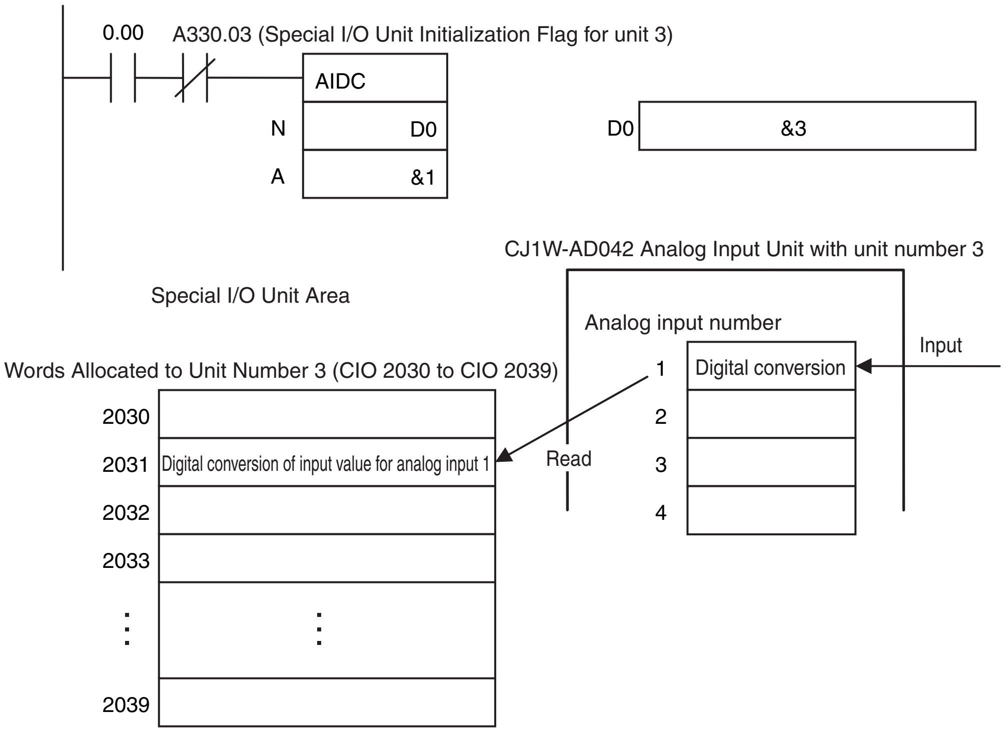
- Use AIDC(216) only for a CJ1W-AD042 Analog Input Unit. If it is executed for any other Special I/O Unit, the instruction execution time may be approx. 1 ms or the Unit may stop operating (i.e., the ERH indicator will light).
- The specified CJ1W-AD042 Analog Input Unit must be placed in Direct Conversion Mode before AIDC(216) is executed. If it is not in Direct Conversion Mode, the instruction will not be executed and the Error Flag will turn ON.
- Do not execute this instruction in both a cyclic task and an interrupt task. If it is executed both in a cyclic task and an interrupt task, the instruction in the interrupt task will not be executed and the Error Flag will turn ON.

- If an attempt is made to execute this instruction while the CJ1W-AD042 Analog Input Unit is being initialized, it will not be executed and the Error Flag will turn ON. To ensure that the Unit is not being initialized when this instruction is executed, use the High-speed I/O Unit Initialization Flag in a normally closed condition for the instruction.



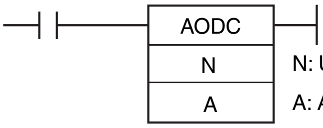
Example Programming

When CIO 0.00 is ON in the following example, the digital conversion of the input value of analog input 1 is read from the CJ1W-AD042 Analog Input Unit with unit number 3 and stored in CIO 2031 (in the words allocated to Special I/O Unit with unit number 3 in the Special I/O Unit Area).



AODC

Instruction	Mnemonic	Variations	Function code	Function
ANALOG OUTPUT DIRECT CONVERSION	AODC	@AODC	217	Outputs the output set value for the specified analog output number to the CJ1W-DA042V Analog Output Unit in Direct Conversion Mode.

	AODC
Symbol	 N: Unit number A: Analog output number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Unit number	UNIT	1
A	Analog output number	UINT	1

N: Unit number

Specify the unit number between 0 and 95 using &0 to &95 decimal or #0000 to #005F hex.

A: Analog output number

Specify the analog output or outputs using a constant between &0 and &4 decimal or #0000 and #0004 hex.

&0 (#0000): Analog output 1 to the last analog output

&1 (#0001): Analog output 1

&2 (#0002): Analog output 2

&3 (#0003): Analog output 3

&4 (#0004): Analog output 4

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
A	---	---	---	---	---	---	---	---	---	---	OK	---		---				

Flags

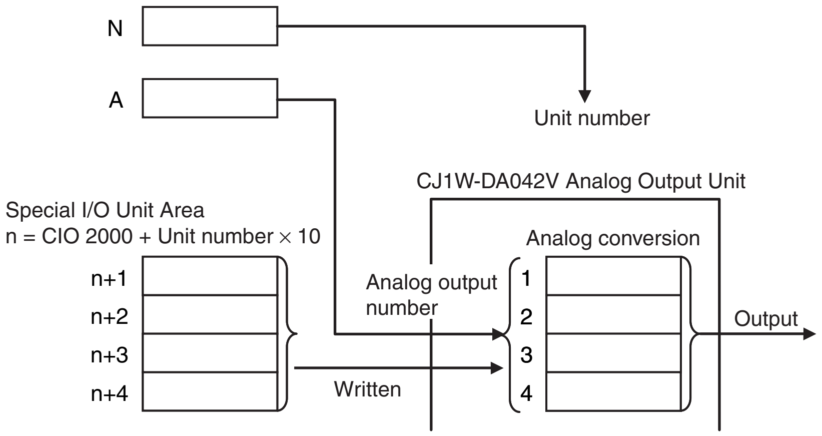
Name	Label	Operation
Error Flag	ER	• ON if the unit number in N is not between &0 and &95 decimal (#0000 and #005F hex). • ON if the specified unit is not a CJ1W-DA042V Analog Output Unit. • ON if the specified CJ1W-DA042V Analog Output Unit is being initialized or restarted. • ON if there is no Unit with the unit number specified in N. • ON if AODC(217) is executed in an interrupt task when it is also being executed in a cyclic task. • ON if an analog output that is not being used is specified in A. • ON if the specified CJ1W-DA042V Analog Output Unit is not in Direct Conversion Mode. • ON if the output SV in the Special I/O Unit Area for the specified analog output is out of range. If outputting all analog outputs is specified in A, the Error Flag will turn ON if the output SV for even one of them is out of range. • ON if there is a DM parameter setting error for specified CJ1W-DA042V Analog Output Unit. (The ERC indicator will be lit.) • OFF in all other cases.
Equals Flag	=	• ON if output the data was completed normally. • ON if outputting the data was not completed normally.

Related Auxiliary Area Words and Bits

Name	Addresses	Operation
Special I/O Unit Initialization Flags	A330.00 to A335.15	ON when a Special I/O Unit is being initialized. OFF: Not being initialized. ON: Being initialized.

Function

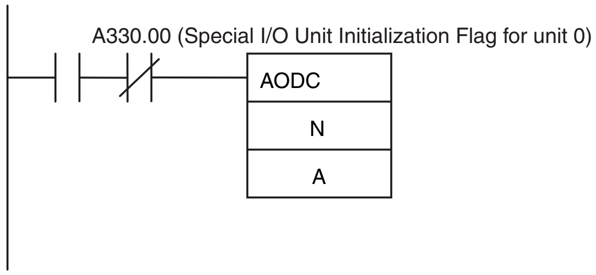
AODC(217) executes analog conversion for the CJ1W-DA042V Analog Output Unit with the unit number specified in N and outputs the conversion results immediately to the analog output or outputs specified in A. If 0 is specified for A, four words of data are stored. If 1 to 4 is specified for A, one word of data is stored. The specified CJ1W-DA042V Analog Output Unit must be in Direct Conversion Mode before AODC(217) is executed. Refer to the *SYSMAC CS/CJ-series Analog I/O Units Operation Manual* (Cat. No. W345) for details on the Direct Conversion Mode and the number of analog outputs.



Precaution

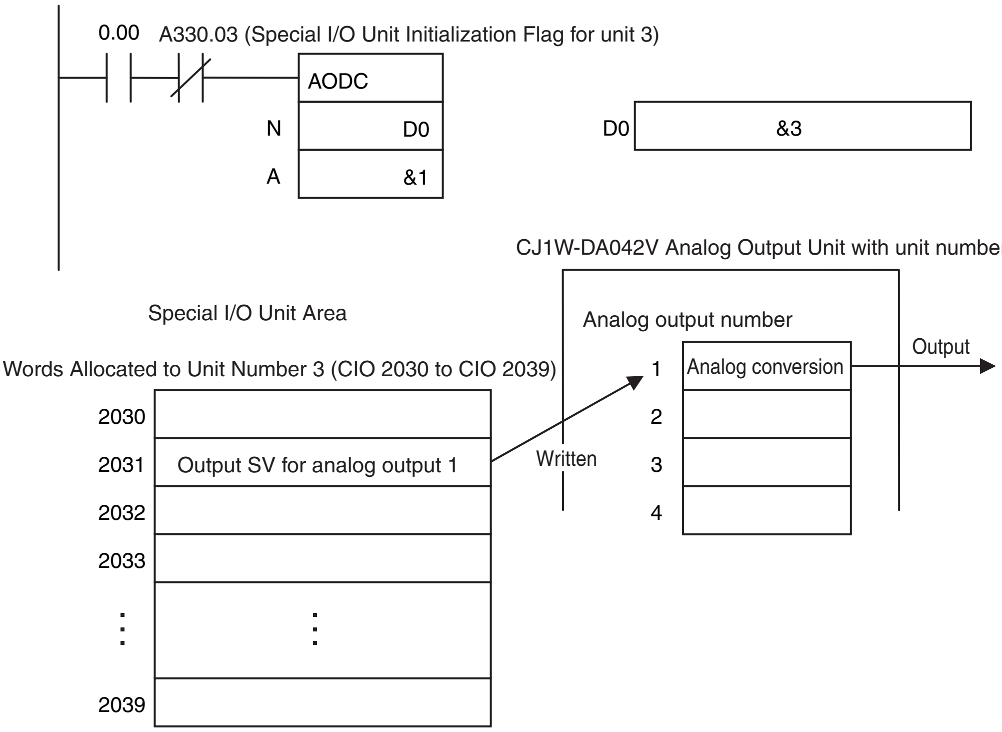
- Use AODC(217) only for a CJ1W-DA042V Analog Output Unit. If it is executed for any other Special I/O Unit, the instruction execution time may be approx. 1 ms or the Unit may stop operating (i.e., the ERH indicator will light).
- The specified CJ1W-DA042V Analog Output Unit must be placed in Direct Conversion Mode before AODC(217) is executed. If it is not in Direct Conversion Mode, the instruction will not be executed and the Error Flag will turn ON.
- Do not execute this instruction in both a cyclic task and an interrupt task. If it is executed both in a cyclic task and an interrupt task, the instruction in the interrupt task will not be executed and the Error Flag will turn ON.

- If an attempt is made to execute this instruction while the CJ1W-DA042V Analog Output Unit is being initialized, it will not be executed and the Error Flag will turn ON. To ensure that the Unit is not being initialized when this instruction is executed, use the High-speed I/O Unit Initialization Flag in a normally closed condition for the instruction.



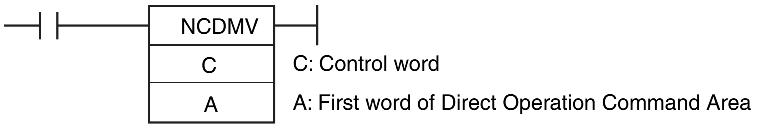
Example Programming

When CIO 0.00 is ON in the following example, the output SV for analog output 1 stored in CIO 2031 (in the words allocated to Special I/O Unit with unit number 3 in the Special I/O Unit Area) is output from analog output 1 of the CJ1W-DA042V Analog Output Unit with unit number 3.



NCDMV

Instruction	Mnemonic	Variations	Function code	Function
PCU HIGH-SPEED POSITIONING	NCDMV	@NCDMV	218	NCDMV(218) starts absolute or relative high-speed point-to-point positioning for the specified axis of a CJ1W-NC□□4, CJ1W-NC□81, or CJ1W-NC□82 Position Control Unit.

	NCDMV
Symbol	

Applicable Program Areas

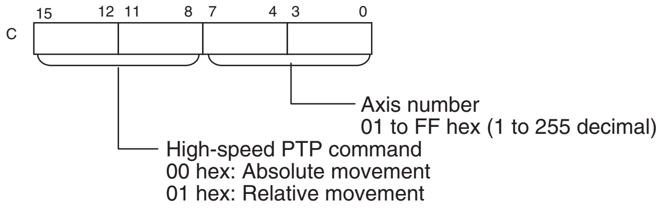
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	DWORD	2
A	First word of Direct Operation Command Area	WORD	Variable

C: Control Data

C: High-speed PTP Command and Axis Number



C+1: Position Control Unit

- CJ1W-NC□□4 (Special I/O Unit)
  0000 to 005F hex (unit number 0 to 95)
- CJ1W-NC□81/NC□82 (CPU Bus Unit)
  8000 to 800F hex (unit number 0 to F)

A: First Word of Direct Operation Command Area

Specify the first word of the Direct Operation Command Area that is set in the common parameters of the Position Control Unit.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
A	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK		OK				

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if a Unit with the number C+1 is not in range. • ON if the Unit specified in C+1 is not a CJ1W-NC214, CJ1W-NC234, CJ1W-NC414, CJ1W-NC434, CJ1W-NC281, CJ1W-NC481, CJ1W-NC881, CJ1W-NCF81, CJ1W-NC482, or CJ1W-NC882 Position Control Unit. • ON if a Unit with the number C+1 is not in the PLC. • ON if the axis number in C is 00. • OFF in all other cases.
Equals Flag	P_EQ	• ON if positioning was started normally by the Position Control Unit. • OFF if the high-speed PTP command in C is not allowed for the specified Position Control Unit. • OFF if the axis number in C is not allowed for the specified Position Control Unit. • OFF if the axis number in C is disabled for the specified Position Control Unit. • OFF if NCDMV(218) is executed for a Position Control Unit that is not ready.

Related Auxiliary Area Words and Bits

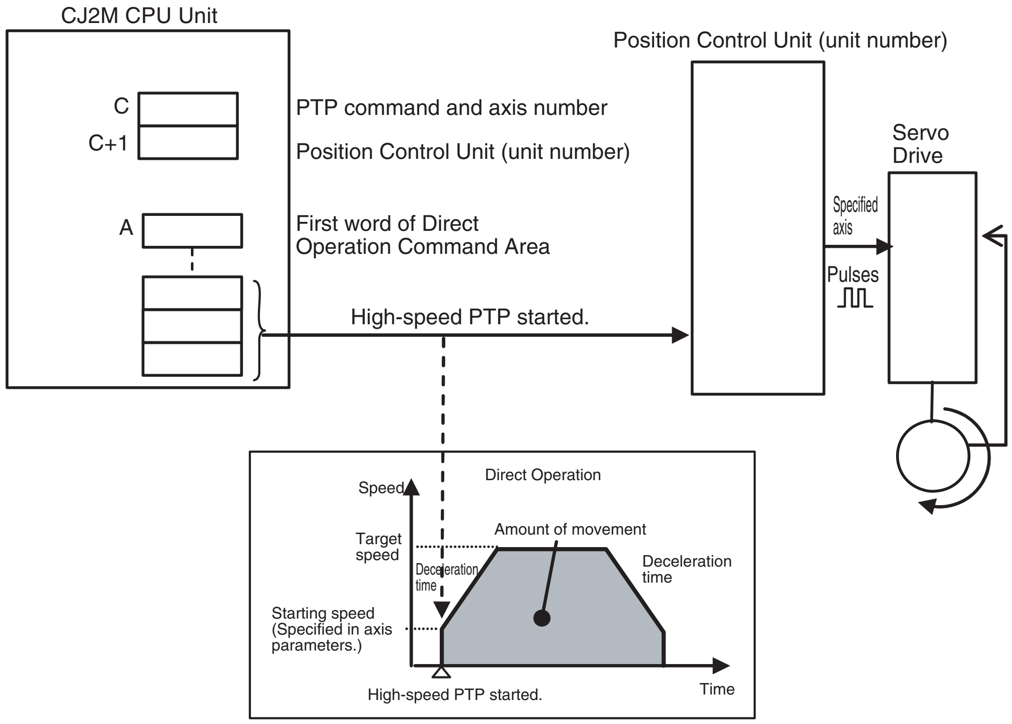
Name	Addresses	Operation
Special I/O Unit Initialization Flags	A330.00 to A335.15	Turns ON when the Special I/O Unit is being initialized. OFF: Not initializing. ON: Initializing.
CPU Bus Unit Initializing Flags	A302.00 to A302.15	Turns ON when the CPU Bus Unit is being initialized. OFF: Not initializing. ON: Initializing.

Function

NCDMV(218) starts absolute or relative high-speed point-to-point positioning for the axis specified in C of the Position Control Unit specified in C+1.

Positioning can be started for only one axis at a time with each NCDMV(218) instruction.

The Equals Flag in the Conditions Flags is turned ON if positioning is started normally in the Position Control Unit.



The following Position Control Units can be specified.

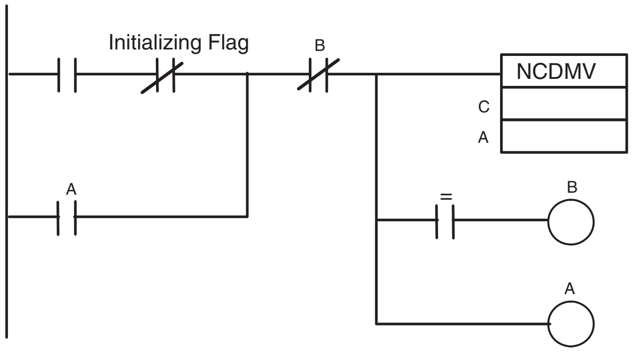
CJ1W-NC214, CJ1W-NC234, CJ1W-NC414, and CJ1W-NC434 (Special I/O Units)

CJ1W-NC281, CJ1W-NC481, CJ1W-NC881, CJ1W-NCF81, CJ1W-NC482, or CJ1W-NC882 (CPU Bus Units)

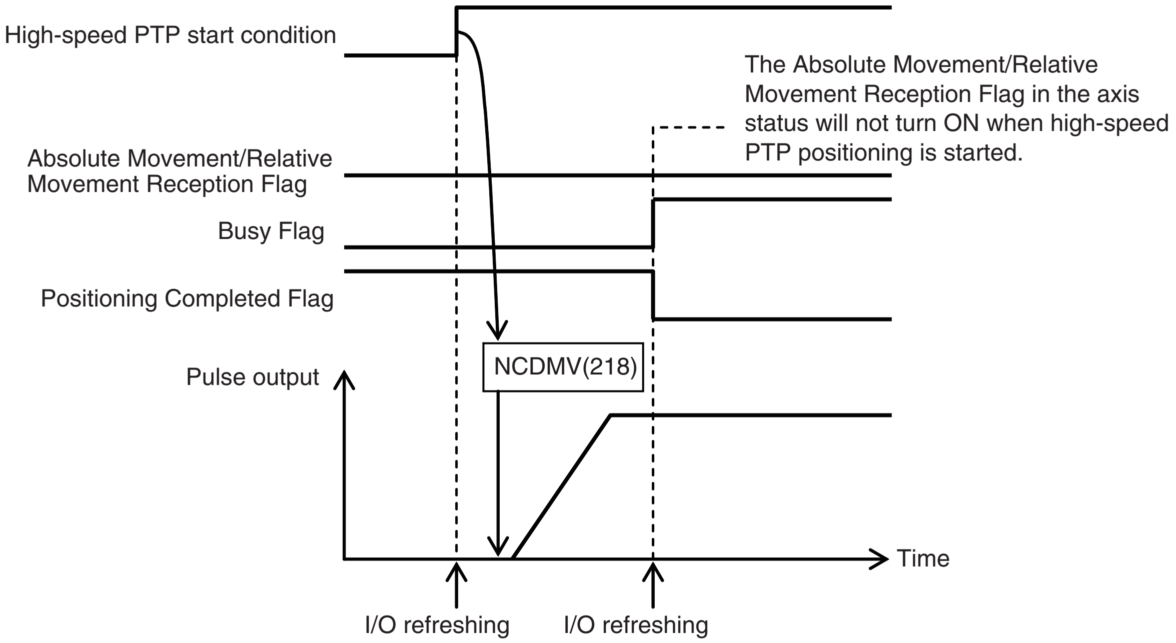
Refer to the *CJ-series Position Control Unit Operation Manual* (Cat. No. W426) for details on using direct operation.

Precautions for Correct Use

- The Position Control Unit can execute only one command for one NCDMV(218) instruction each control cycle. If high-speed PTP movements are started continuously, the Position Control Unit may not be able to accept all of them. (In this case, the Equals Flag will be OFF.) Use the Equals Flag to create a self-maintaining program, as shown below, so that NCDMV(218) will not be executed until the Equals Flag turns ON, as shown below.
- NCDMV(218) will not be executed when the Position Control Unit is being initialized. Use the following Unit Initializing Flags in NC conditions as execution conditions for NCDMV(218). (The Unit Initializing Flags are A330.00 to A335.15 for Special I/O Units and A302.00 to A302.15 for CPU Bus Units.)



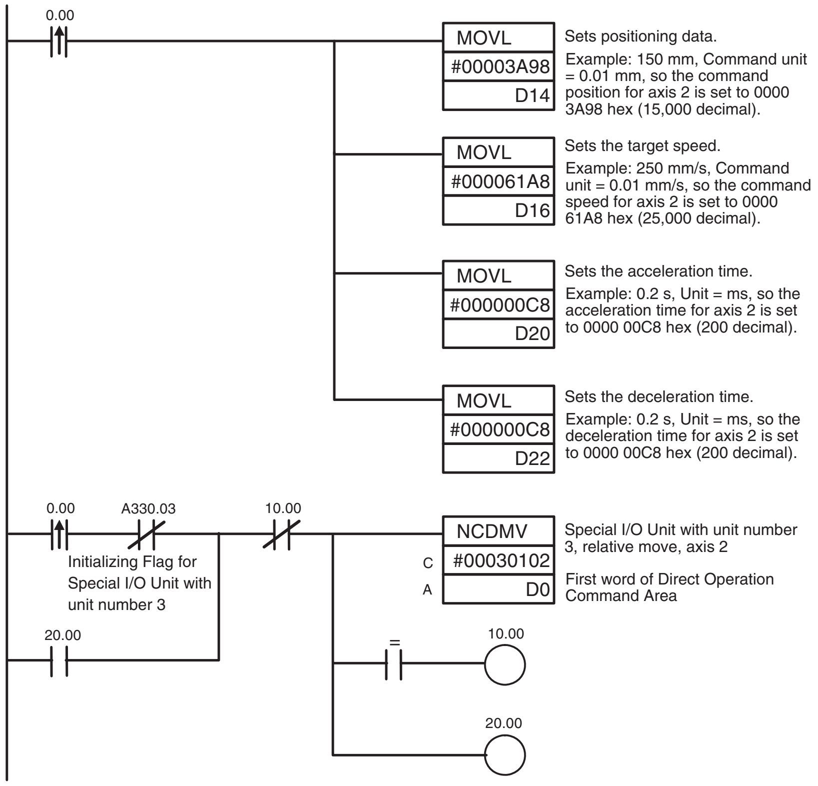
- When starting high-speed PTP positioning, pulse output will start as soon as NCDMV(218) is executed. Updating axis status, such as the Axis Busy Flag, will be performed in the next I/O refresh period.

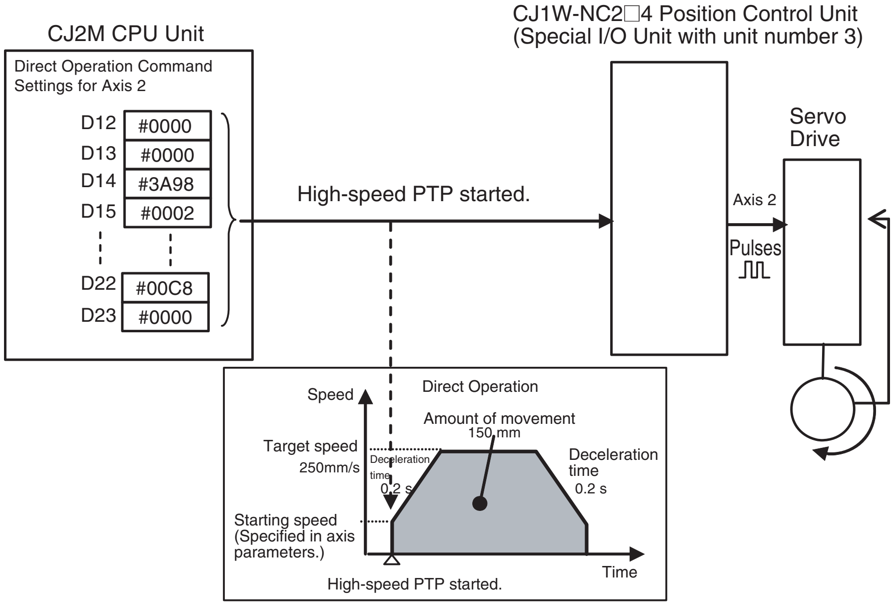


- NCDMV(218) can be used in an interrupt task, which allows high-speed PTP positioning to be started with an interrupt. However, do not execute NCDMV(218) in a cyclic task for the same Unit if NCDMV(218) is executed in an interrupt task. (The same thing applies to IORD(222) and IOWR(223).) If an attempt is made to execute NCDMV(218) in an interrupt task when it is already being executed for the same Unit in a cyclic task, it will cause a duplicate refresh error (non-fatal) and A402.13 will turn ON.

Example Programming

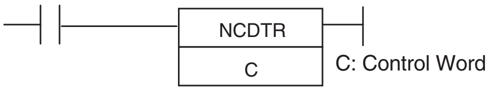
When CIO 0.00 turns ON in the following example, an relative movement is executed using direct operation for axis 2 of the CJ1W-NC2□4 Position Control Unit (a Special I/O Unit) with unit number 3. If the first word of the Direct Operation Command Area is D0, the direct operation command settings in D12 to D23 are used to start high-speed PTP positioning.





NCDTR

Instruction	Mnemonic	Variations	Function code	Function
PCU POSITIONING TRIGGER	NCDTR	@NCDTR	219	NCDTR(219) is used to start a sequence for Memory Operation of a CJ1W-NC□81/NC□82 Position Control Unit when the start condition for the sequence is waiting for a command from NCDTR(219).

	NCDMV
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control data	DWORD	2

C: Control data

C: Task number in Position Control Unit

Specify the task that is set in the Memory Operation Parameters of the Position Control Unit between 0001 and 00FF hex (1 to 255 decimal).

C+1: Position Control Unit

Specify 8000 to 800F hex (unit number 0 to F).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if a Unit with the number C+1 is not in range. • ON if the Unit specified in C+1 is not a CJ1W-NC281, CJ1W-NC481, CJ1W-NC881, CJ1W-NCF81, CJ1W-NC482, or CJ1W-NC882 Position Control Unit. • ON if there is no Unit with the unit number specified in C+1. • OFF in all other cases.
Equals Flag	P_EQ	• ON if positioning was started normally by the Position Control Unit. • OFF if the task number in C is out of range for the specified Position Control Unit. • OFF if the task number in C is not waiting to be started by NCDTR(219) for the specified Position Control Unit. • OFF if NCDTR(219) is executed for a Position Control Unit that is not ready.

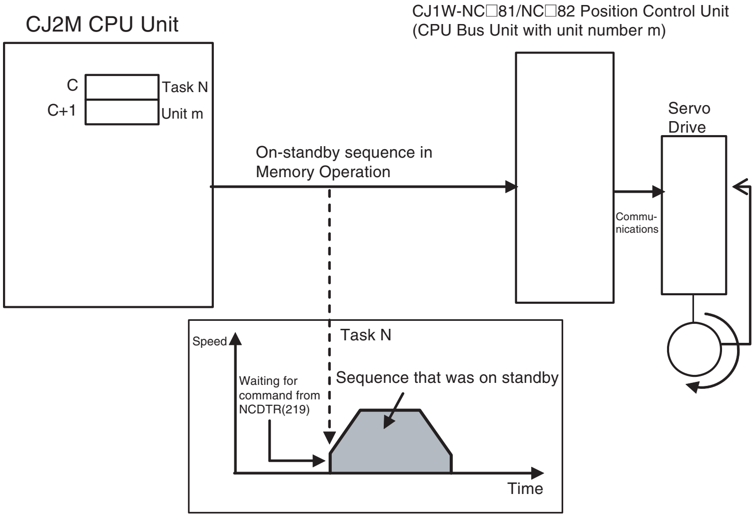
Related Auxiliary Area Words and Bits

Name	Addresses	Operation
CPU Bus Unit Initializing Flags	A302.00 to A302.15	Turns ON when the CPU Bus Unit is being initialized. OFF: Not initializing. ON: Initializing.

Function

NCDTR(219) starts a sequence that is waiting for a command input from NCDTR(219) for the task specified in C for Memory Operation in the Position Control Unit specified in C+1.

The Equals Flag in the Conditions Flags is turned ON if the sequence is started normally in the Position Control Unit.

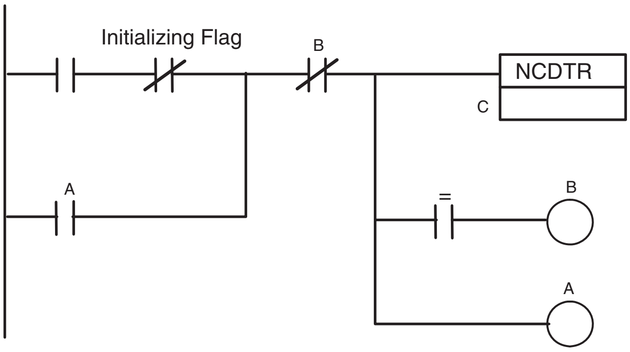


The following Position Control Units can be specified.
CJ1W-NC281, CJ1W-NC481, CJ1W-NC881, CJ1W-NCF81, CJ1W-NC482, and CJ1W-NC882 (CPU Bus Units)

Refer to the *CJ-series Position Control Unit Operation Manual* (Cat. No. W426) for specifications on Memory Operation.

Precaution

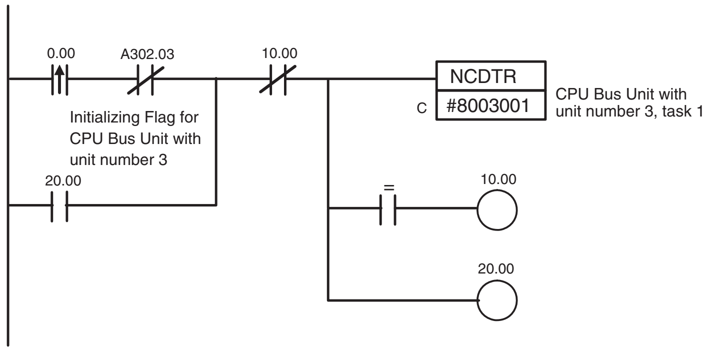
- The Position Control Unit can execute only one command for one NCDTR(219) instruction each control cycle.
  If NCDTR(219) is executed continuously for different tasks, the Position Control Unit may not be able to accept all of them. (In this case, the Equals Flag will be OFF.)
  Use the Equals Flag to create a self-maintaining program, as shown below, so that NCDTR(219) will not be executed until the Equals Flag turns ON, as shown below.
- NCDTR(219) will not be executed when the Position Control Unit is being initialized.
  Use the following Unit Initializing Flags in NC conditions as execution conditions for NCDTR(219). (The Unit Initializing Flags are A302.00 to A302.15 for CPU Bus Units.)

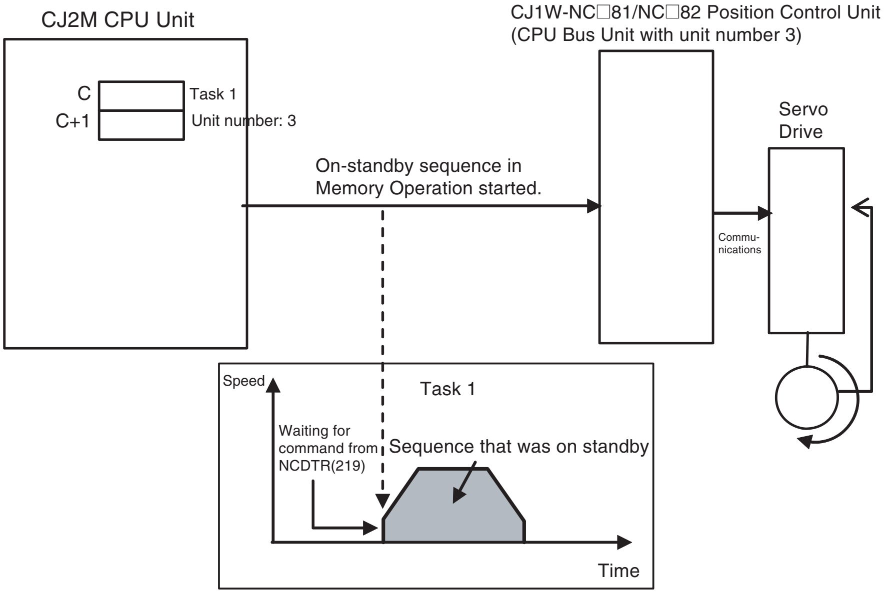


- NCDTR(219) can be used in interrupt tasks.
  However, do not execute NCDTR(219) in a cyclic task for the same Unit if NCDTR(219) is executed in an interrupt task. (The same thing applies to IORD(222) and IOWR(223).)
  If an attempt is made to execute NCDTR(219) in an interrupt task when it is already being executed for the same Unit in a cyclic task, it will cause a duplicate refresh error (non-fatal) and A402.13 will turn ON.

Example Programming

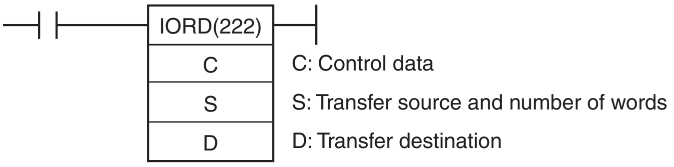
When CIO 0.00 turns ON in the following example, the sequence that is waiting for a command input from NCDTR(219) is started for task 1 of the Position Control Unit with unit number 3.





IORD

Instruction	Mnemonic	Variations	Function code	Function
INTELLIGENT I/O READ	IORD	@IORD	222	Reads the contents of memory area of a Special I/O Unit or CPU Bus Unit.

Symbol	IORD
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control data	UINT	1
S	Transfer source and number of words	UDINT	2
D	Transfer destination	UINT	Variable

Note If IORD(222) is executed for a CPU Bus Unit running under a CPU Unit that does not support using IORD(222) for CPU Bus Units, an error will occur and the ER Flag will turn ON.

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

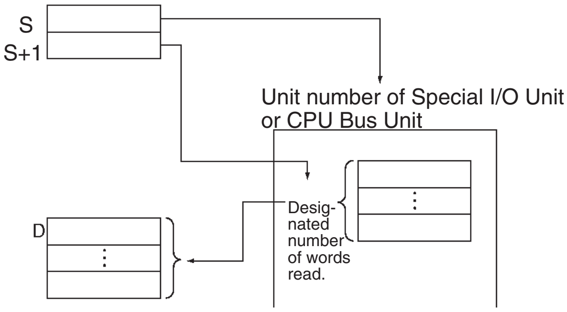
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S												---						
D											---							

Flags

Name	Label	Operation
Error Flag	P_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 the designated Special I/O Unit is on SYSMAC BUS. • ON if a Special I/O Unit or CPU Bus Unit not affected by IORD(222) is designated. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.
Equals Flag	P_EQ	• ON if reading operation is completed normally. • OFF if reading operation is not completed normally.

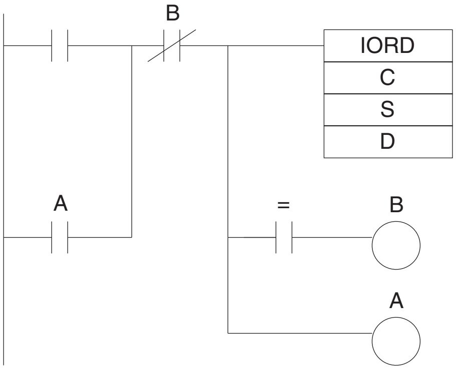
Function

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. Only Special I/O Units or CPU Bus Units mounted on CPU Racks or Expansion I/O Racks can be designated. 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.



Hint

- 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.



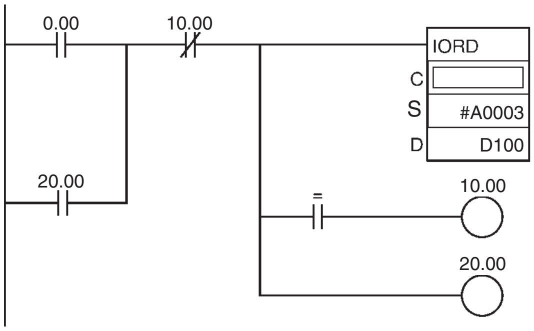
- When the input condition is met, self maintenance is performed by output A and IORD(222) is executed with each cycle until the Equals Flag turns ON. When the reading is completed and the Equals Flag turns ON, output B turns ON and the self maintenance is cleared.
- Be sure to place condition flags 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.

Precaution

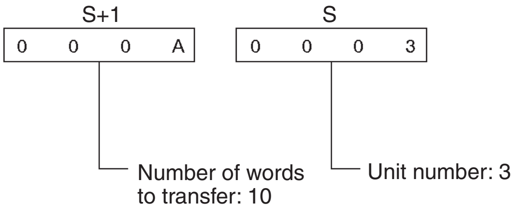
- IORD(222) can be used in an interrupt task, which allows high-speed processing of specific I/O data with an interrupt. If IORD(222) is used in an interrupt task, always disable cyclic refreshing of the specified Special I/O Unit by turning ON the corresponding Special I/O Unit Cyclic Refreshing Disable Bit in the PLC Setup.
- When cyclic refreshing of the specified Special I/O Unit is enabled in the PLC Setup (the corresponding Special I/O Unit Cyclic Refreshing Disable Bit is OFF), a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A402.13) will go ON in the following cases.
    - Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225).
    - Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).

Example Programming

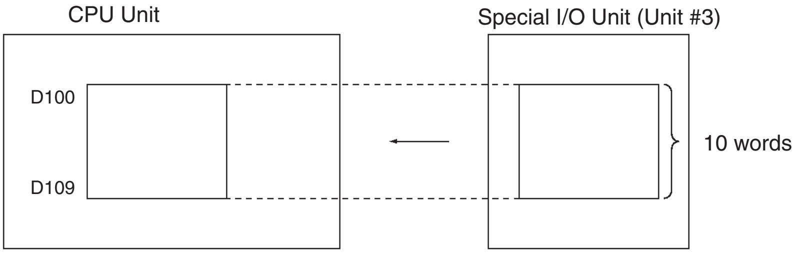
In this example, IORD(222) is used to read data.



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.

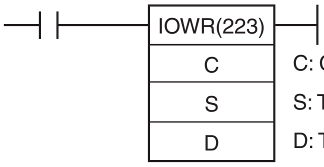


The control code (C) varies depending on the Special I/O Unit.



IOWR

Instruction	Mnemonic	Variations	Function code	Function
INTELLIGENT I/O WRITE	IOWR	@IOWR	223	Outputs the contents of the CPU Unit's I/O memory area to a Special I/O Unit or CPU Bus Unit.

	IOWR
Symbol	 C: Control data S: Transfer source and number of words D: Transfer destination and number of words

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

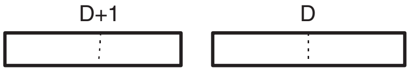
Operand	Description	Data type	Size
C	Control data	UINT	1
S	Transfer source and number of words	WORD	Variable
D	Transfer destination and number of words	UDINT	2

Note If IOWR(223) is executed for a CPU Bus Unit running under a CPU Unit that does not support using IOWR(223) for CPU Bus Units, an error will occur and the ER Flag will turn ON.

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)



D+1: Leftmost 4 digits
D: Rightmost 4 digits

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S												---						
D																		

Flags

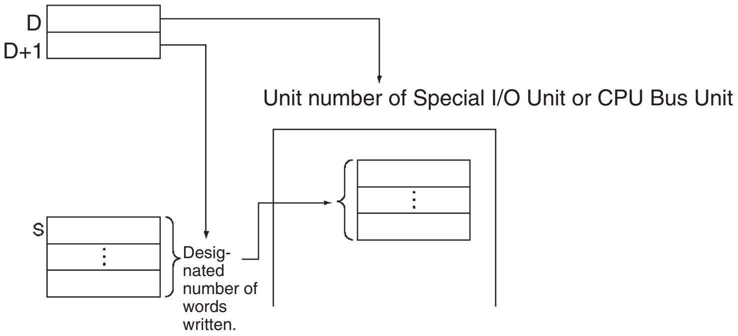
Name	Label	Operation
Error Flag	P_ER	• ON if the number of words to transfer (D) is outside the range of 0001 to 0080 hex. • ON if the unit number (D) is outside the range of 0000 to 005F hex or 8000 to 800F hex. • ON if S is designated by a constant when the number of words to be transferred (D+1) is not 0001 hex. • ON if the designated Special I/O Unit is on SYSMAC BUS. • ON if a Special I/O Unit or CPU Bus Unit not affected by IOWR(223) is designated. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.
Equals Flag	P_EQ	• ON if writing operation is completed normally. • OFF if writing operation is not completed normally.

Note
- 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 OFF if the writing operation cannot be completed normally due to the Special I/O Unit or CPU Bus Unit being busy.
- An error will occur if there is an I/O Unit verification error, Special I/O Unit setting error, CPU Bus Unit setting error, Special I/O Unit error, or CPU Bus Unit error in a Special I/O Unit or CPU Bus Unit.

Function

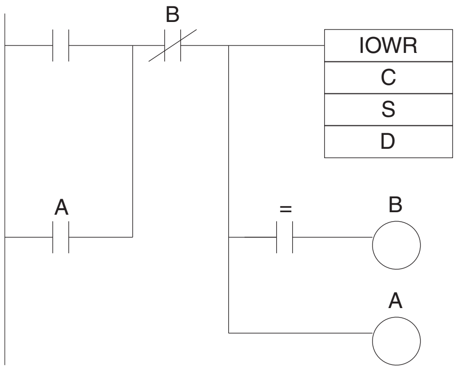
IOWR(223) writes the designated number of words (D) from the first source word (designated by S) onwards and outputs them to the Special I/O Unit or CPU Bus Unit that has the unit number designated by D. Only Special I/O Units or CPU Bus Units mounted on CPU Racks or Expansion I/O Racks can be designated.

For details on using the IOWR instruction, see the manual of the special I/O unit (or CPU special unit) to be written to.



Hint

- When IOWR(223) is executed, the execution results are reflected in the condition flags. In particular, the Equals Flag turns ON when writing 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.



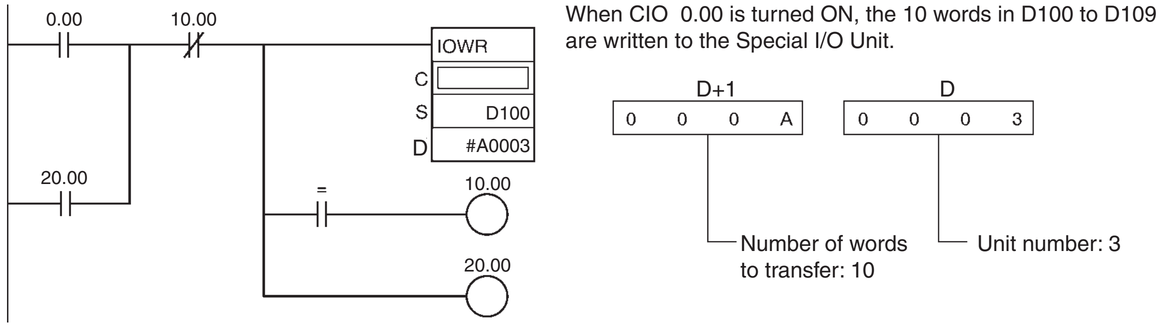
- When the input condition is met, self maintenance is performed by output A and IOWR(223) is executed with each cycle until the Equals Flag turns ON. When the writing is completed and the Equals Flag turns ON, output B turns ON and the self maintenance is cleared.
- Be sure to place condition flags 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.

Precaution

- IOWR(223) can be used in an interrupt task, which allows high-speed processing of specific I/O data with an interrupt. If IOWR(223) is used in an interrupt task, always disable cyclic refreshing of the specified Special I/O Unit by turning ON the corresponding Special I/O Unit Cyclic Refreshing Disable Bit in the PLC Setup.
- When cyclic refreshing of the specified Special I/O Unit is enabled in the PLC Setup (the corresponding Special I/O Unit Cyclic Refreshing Disable Bit is OFF), a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A40213) will go ON in the following cases.
  - Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225).
  - Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).

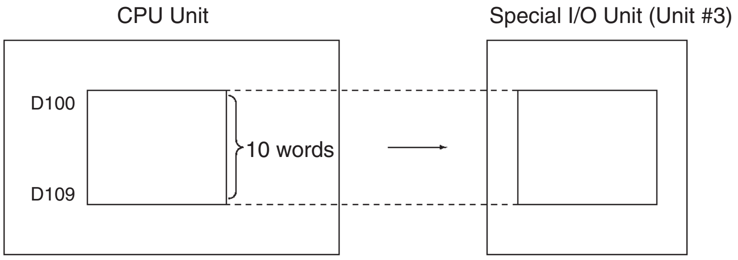
Example Programming

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.

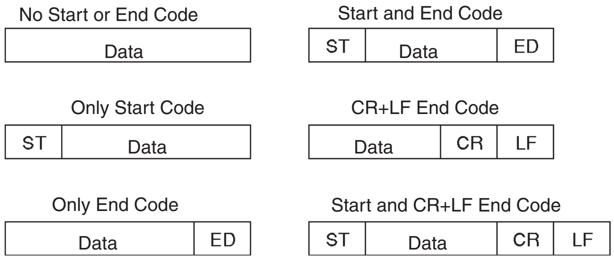
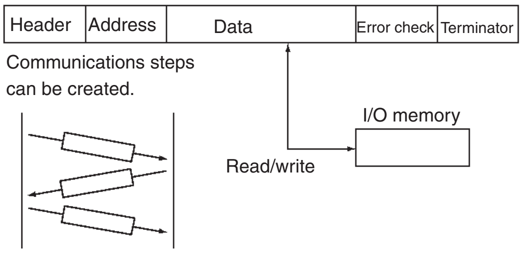


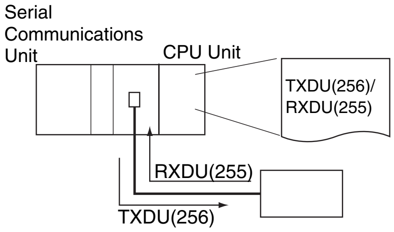
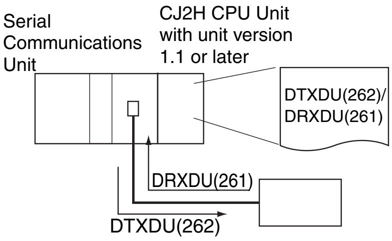
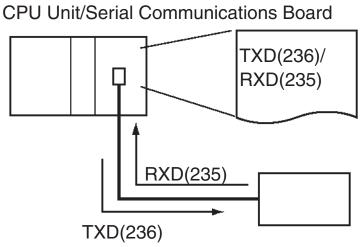
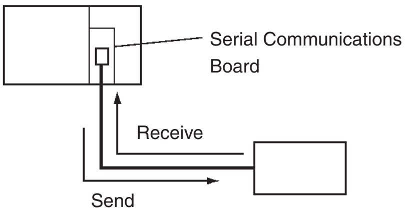
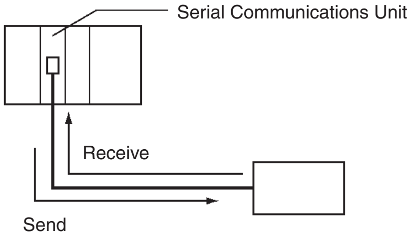
Serial Communications Instructions

Serial Communications Instructions

There are two types of serial communications instruction. The TXD(236), RXD(235), TXDU(256), RXDU(255), DTXDU(262), and DRXDU(261) instructions send and receive data in no-protocol (custom) communications with an external device. PMCR(260)/PMCR2(264) sends and receives data using user-defined protocols with an external device. The difference is shown in the following tables.

Note
- TXD(236) and RXD(235) are used for built-in CPU Unit serial ports and serial ports on Serial Communications Boards with unit version 1.2 or later.
- TXDU(256) and RXDU(255) are used for serial ports on Serial Communications Units with unit version 1.2 or later.
- DTXDU(262) and DRXDU(261) are used for serial ports on CJ1W-SCU22/SCU32/SCU42 Serial Communications Units mounted to CJ2H CPU Units with unit version 1.1 or later. These instructions enable high-speed data communications.
- PMCR(260) and PMCR2(264) can be used only for the serial ports on Serial Communications Units/Boards.

Instructions	Communications frames	Function
TXD(236), RXD(235), TXDU(256), RXDU(255), DTXDU(262), and DRXDU(261)	Any of the following can be used.  	Sends or receives data in one direction only. A send delay can be set.
PMCR(260) and PMCR2(264)	The following type of frames (messages) can be created to meet the requirements of the external device.  	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
TXDU(256) and RXDU(255)	No-protocol (custom)	Serial Port of Serial Communications Unit with Unit Version 1.2 or Later 
DTXDU(262) and DRXDU(261)	No-protocol (custom)	Serial Port of CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit 
TXD(236) and RXD(235)	No-protocol (custom)	Serial Port in CPU Unit or Serial Communications Board with Unit Version 1.2 or Later 
PMCR(260) and PMCR2(264)	Protocol macro	Serial Communications Board (CS Series only)  Serial Communications Unit 

● Communications Port Allocations

Although the communications port (see note) was specified in the operands of previous Serial Communications Instructions, the Network Communications Instructions use automatic port allocation with new communications ports (8 to 71) to enable executing up to 64 of the Network Communications Instruction simultaneously. With these instructions, there is no need for the user to keep track of the communications ports.

Each time one of these instructions is executed, the number of ports stored in A211 (Number of Ports Available) will be decremented. Each time a response is returned for a communications port, the value in A211 will be incremented. This system enables monitoring communications performance.

Communications ports	Application	Assignment method
0 to 7	Network Communications Instructions, Serial Communications Instructions, easy backup (SEND, CMND, PMCR, TXDU, etc.)	Specified in operand. 0 to 7: The specified communications port is assigned. F: The communications port is automatically assigned between 0 and 7.
8 to 71	New Serial Communications Instructions (PMCR2)	Automatically assigned, with the number of available ports stored in A211.

Note The communications ports are internal logical ports (virtual ports) used to manage simultaneous execution of communications instructions over multiple cycles (including communications processing, file handling, etc.).

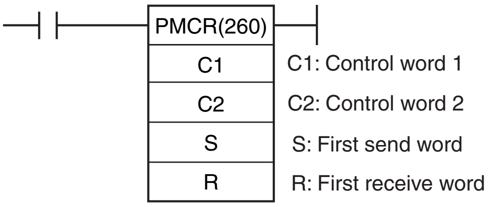
● Auxiliary Area Words and Bits

The Network Communications Instructions use the communications information in operand I in place of the Communications Port Enabled Flags in A202 used by previous instructions.

Auxiliary Area	Network Communications Instructions for CJ1/CS1/CJ2	CJ2 Network Communications Instructions
A202.00 to A202.07	Communications Port Enabled Flag (communications ports 0 to 7)	Instruction operand I Bit 0 (Turns ON when execution has started, and not when execution is enabled.)
A202.08	Not used.	CJ2 Instructions Enabled Flag (default: ON)
A203 to 210	Communications completion codes for communications ports 0 to 7	Instruction operand I+1
A211	Not used.	Number of ports available for CJ2 Network Communications Instruction (A202.08 turns OFF when the value in A211 reaches 0.)
A214.00 to A214.07	First Cycle Flags after Communications Finished	---
A215.00 to A215.07	First Cycle Flags after Communications Error	---
A216 and A217	Communications Port Completion Code Storage Address	---
A218	Used Communications Port Numbers	---
A219	Communications Port Error Flags	Instruction operand I Bit 1

PMCR

Instruction	Mnemonic	Variations	Function code	Function
PROTOCOL MACRO	PMCR	@PMCR	260	Starts a communications sequence (protocol data) that is registered in a Serial Communications Board (CS Series only) or Serial Communications Unit.

	PMCR
Symbol	

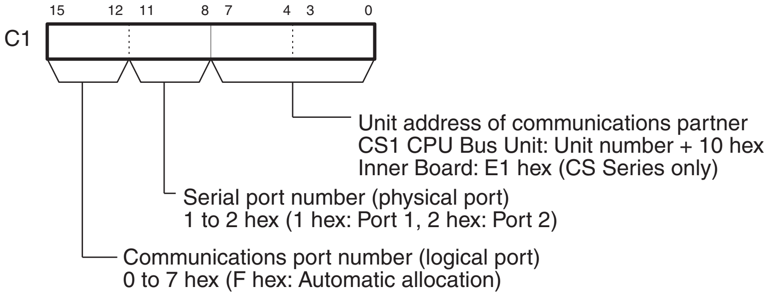
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C1	Control word 1	UINT	1
C2	Control word 2	UINT	1
S	First send word	UINT	Variable
R	First receive word	UINT	Variable

C1: Control word 1



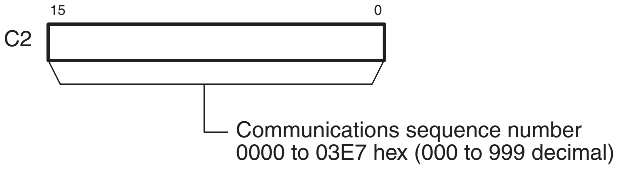
Unit address of communications partner
CS1 CPU Bus Unit: Unit number + 10 hex
Inner Board: E1 hex (CS Series only)

Serial port number (physical port)
1 to 2 hex (1 hex: Port 1, 2 hex: Port 2)

Communications port number (logical port)
0 to 7 hex (F hex: Automatic allocation)

Note Automatic allocation can only be used on CJ2 CPU Units and Lot No. 020601 (manufactured June 1, 2002) or later of CS1-H, CJ1-H, CJ1M, and CS1D (for Single CPU System) CPU Units.

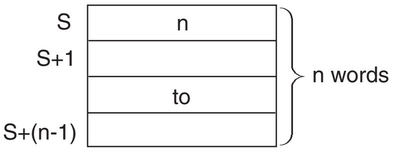
C2: Control word 2



Communications sequence number
0000 to 03E7 hex (000 to 999 decimal)

Send area specification
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.

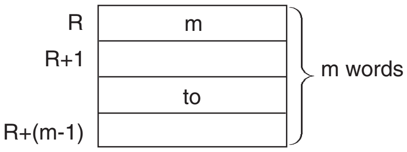
Receive area specification
R: First Receive Word and Receive Area

- Setting Before Executing PMCR

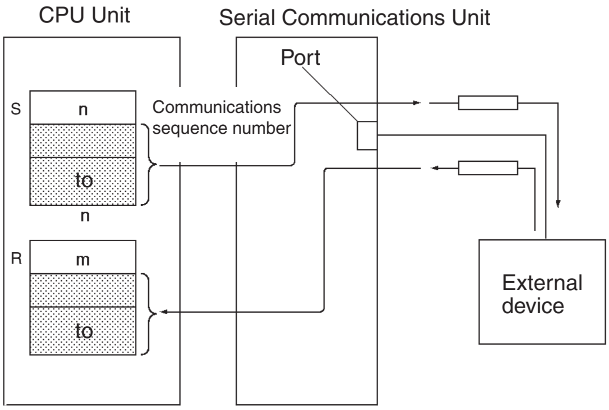
  Set the data specified by m (beginning with D) as the initial data for the receive buffer (backup data for receive failure). Data m can be set to 0002 to 00FA (hex) (2 to 255). If 0000 (hex) or 0001 (hex) is specified for m, the receive area will be cleared to 0 on receive failure.

- When execution of the protocol macro ends

  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).



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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
C1, C2	OK	OK	OK*1	OK	OK	OK	OK	OK*2	OK	OK	OK	OK	---	OK	---	---	---	---
S												---						
R																		

*1 H512 to H1535 cannot be used.

*2 In the EM Area, bank D and higher cannot be used.

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. • ON if the number of words of S or R exceeds 249 (when words are specified). • ON if PMCR(260) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

Related Auxiliary Area Words and Bits

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.

Note Refer to the *CS/CJ-series Communications Commands Reference Manual (W342)* for other response codes.

Inner Board Area (CS Series Only)

Name	Address	Contents
Port 1 Protocol Macro Execution Flag	CIO 1909.15	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	CIO 1919.15	

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	

Function

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. (By means of this function, instead of the present value data being cleared to zero when reception fails, the previously received data is retained.)

Note For information on using automatic allocation (“F”) for the communication port (logical port) number, see “Automatic Allocation of Communications Ports” in “SECTION 3 Instructions” - “Network Instructions”.

Hint

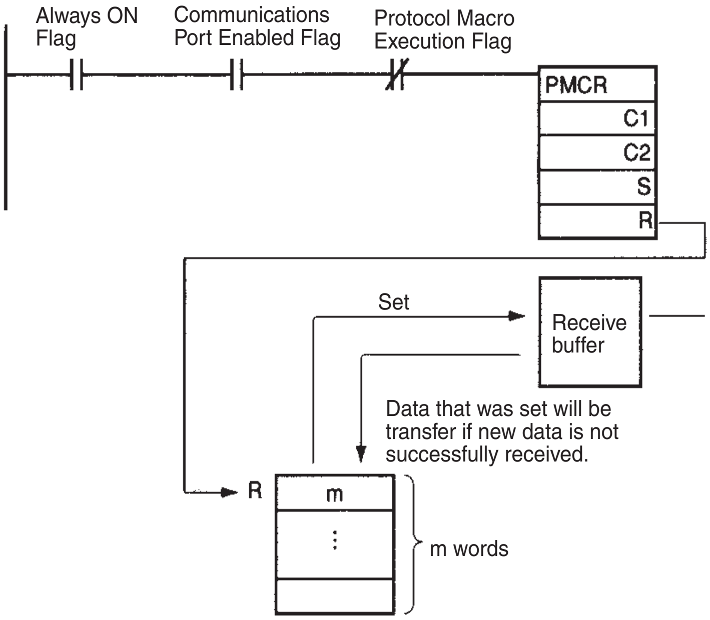
● Holding the Receive Area

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.

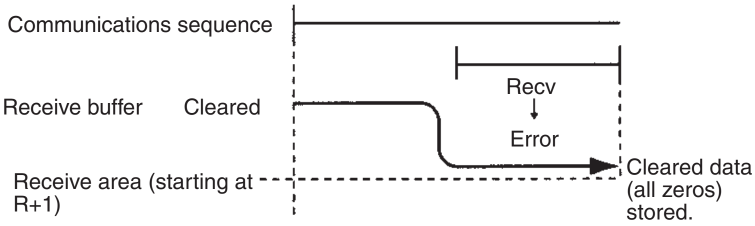
A function is provided to maintain the data in the receive area even when a reception error occurs. If this function is used, data will be transferred from the first m words of the receive area to the receive buffer after the buffer is cleared to all zeros but before the communications sequence is executed. 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.

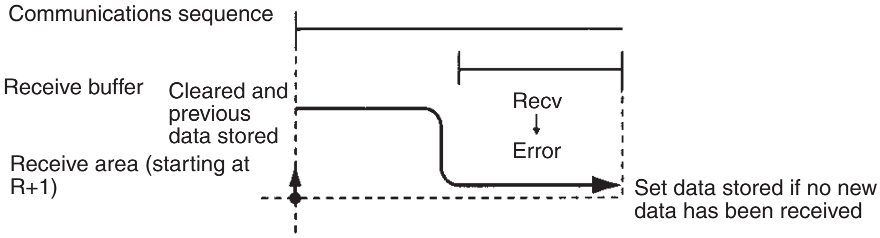
Example: The following programming example shows the instructions used to constantly or periodically execute PMCR(260) to read data through a single receive operation.



Receive Area Not Held



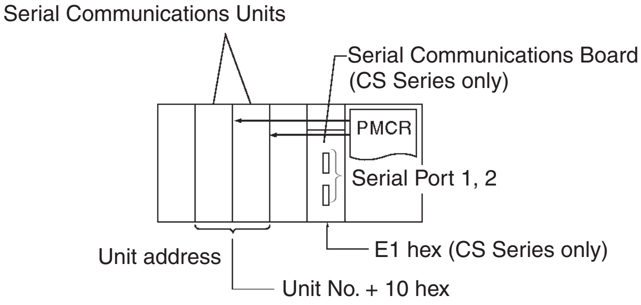
Receive Area Held



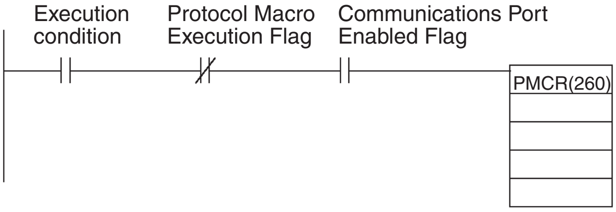
Precautions

- 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 Board (CS Series only) or Serial Communications Unit. Up to 16 Serial Communications Units can be mounted to the CPU Rack and Expansion I/O Racks. The Unit address of the communications partner must be set in bits 0 to 7 of C1 to specify which Unit/Board 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.

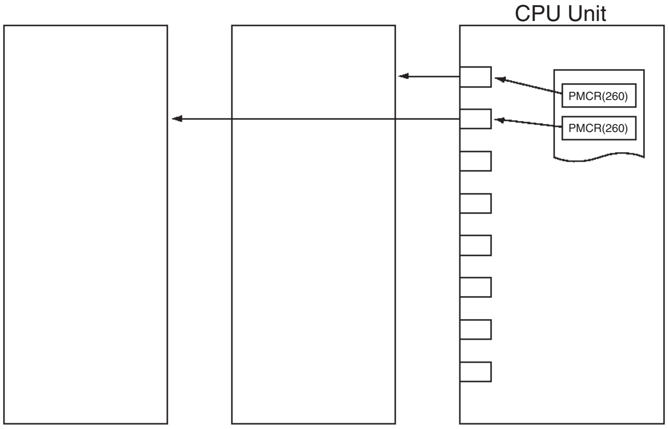
Unit/Board	Unit address
Serial Communications Board (CS Series only)	E1 hex
Serial Communications Unit	Unit number + 10 hex



- 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.

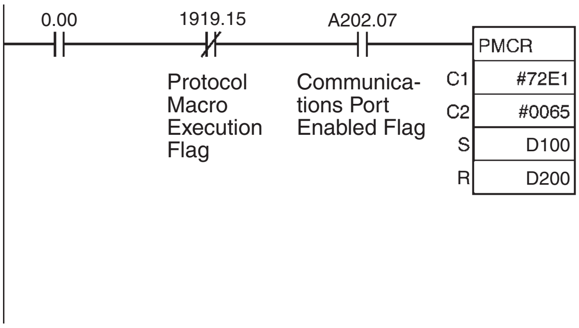


- SEND(090), RECV(098), and CMND(490) also use the logical ports 0 to 7 to execution communications sequences through Serial Communications Unit and Boards (internally using FINS commands). PMCR(260) cannot be executed for a logical port that is already being used by SEND(090), RECV(098), CMND(490)or PMCR(260). To prevent more than one communications sequence from being executed for the same logical port, the corresponding Communications Port Enable Flag (A20200 to A20207) should be used as a NO input in the execution condition for PMCR(260), as shown in the above diagram.



- An error flag will not turn ON when the data of C2 is outside the range. A completion code will be stored in “Network Communications Completion Code” (A203 to A210) of the Auxiliary.

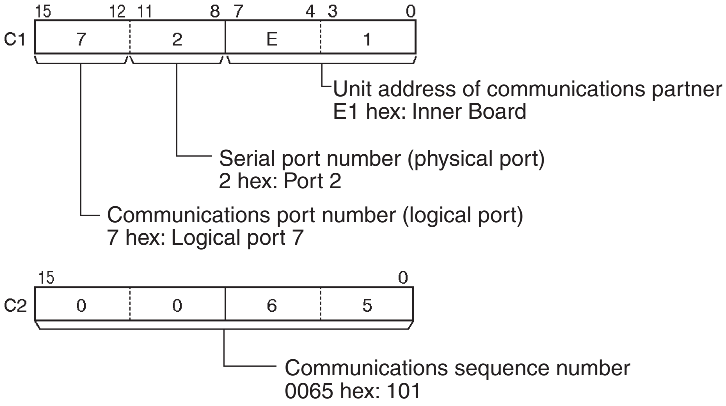
Example Programming

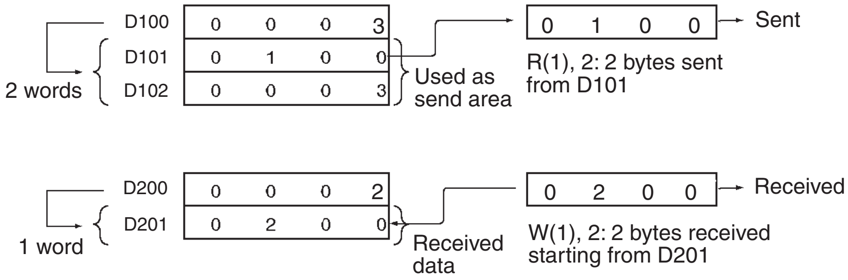


When CIO 0.00 is ON in the following example, communications sequence No. 101 (0065 hex) will be executed as long as the Communications Port Enabled Flag for port 7 (A20207) is ON and the Port 1 Protocol Macro Execution Flag (CIO 1909.15) is OFF.

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.

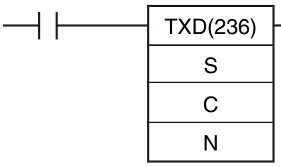




Note As shown above, the symbol read option, R( ), in the send message or the symbol write option, W( ), actually sends/receives data.

TXD

Instruction	Mnemonic	Variations	Function code	Function
TRANSMIT	TXD	@TXD	236	Outputs the specified number of bytes of data from the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports. (The Serial Communications Board must be Ver. 1.2 or later).

	TXD
Symbol	 S: First source word C: Control word N: Number of bytes 0000 to 0100 hex (0 to 256)

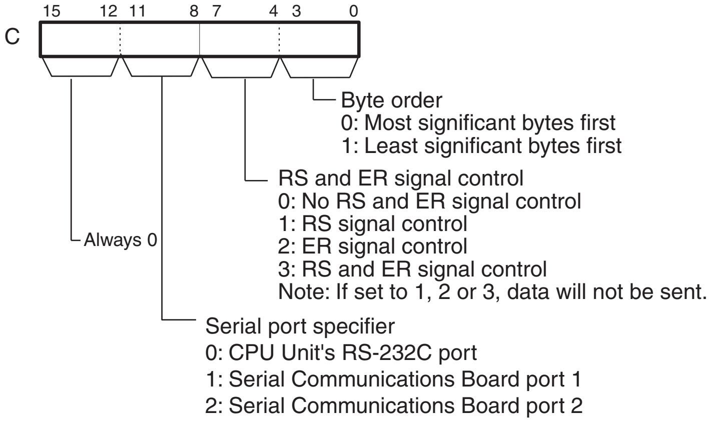
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	UINT	Variable
C	Control word	UINT	1
N	Number of bytes 0000 to 0100 hex (0 to 256)	UINT	1

C: Control word



Byte order
0: Most significant bytes first
1: Least significant bytes first

RS and ER signal control
0: No RS and ER signal control
1: RS signal control
2: ER signal control
3: RS and ER signal control
Note: If set to 1, 2 or 3, data will not be sent.

Serial port specifier
0: CPU Unit's RS-232C port
1: Serial Communications Board port 1
2: Serial Communications Board port 2

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK	OK						
N																		

Flags

Name	Label	Operation
Error Flag	ER	• ON if the CPU Unit's RS-232C port is specified as the send port, but no-protocol mode is not set in the PLC Setup. • ON if one of the Serial Communication Board's serial ports is specified as the send port, but no-protocol mode is not set in the port's allocated DM Setup Area. • 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. (The Send Ready Flag is A39205 for the CPU Unit's RS-232C port, A35605 for Serial Communications Board port 1, or A35613 for Serial Communications Board port 2.) • ON (ER Flag in interrupt tasks) if a TXD(236) or RXD(235) instruction is being executed for the Serial Communications Board in the cyclic task, the cyclic task is interrupted, and another TXD(236) or RXD(235) instruction is executed for the Serial Communications Board in the interrupt task. • ON if a TXD(236) was executed for a serial port on a Serial Communications Board that was being restarted. • OFF in all other cases.

Related Auxiliary Area Words and Bits

● RS-232C incorporated in the CPU Unit

Port	Address	Contents
CPU Bus Unit's built-in RS-232C Port	A392.05	ON when data can be sent in the no-protocol mode.

● Port 1 of Serial Communications Board (Unit Version 1.2 or Later)

Port	Address	Contents
Serial Communications Board port 1	A356.05	ON when data can be sent in the no-protocol mode.

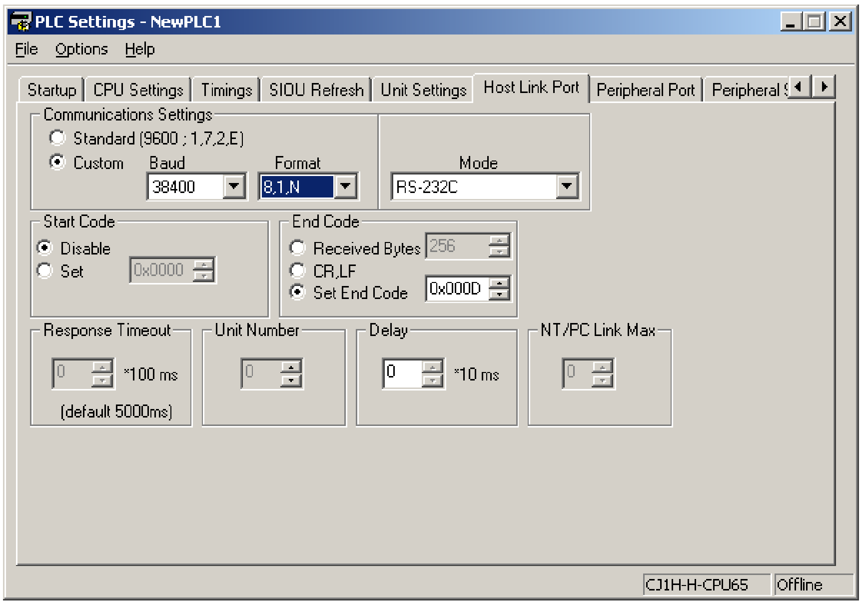
● Port 2

Port	Address	Contents
Serial Communications Board port 2	A356.13	ON when data can be sent in the no-protocol mode.

Name		Address	Contents
Ports 1 and 2	Inner Board Service Disabled Flag	A424.04	ON when TXD(236) is executed for a Serial Communications Board that does not support no-protocol mode (a Board without a version number).

Related PLC Setup Settings for the CPU Unit's Built-in RS-232C Port

● CX Programmer settings



Communications Settings
○ Standard (9600 ; 1,7,2,E)
● Custom
Baud: 38400
Format: 8,1,N
Mode: RS-232C

Start Code
● Disable
○ Set 0x0000

End Code
○ Received Bytes 256
○ CR,LF
● Set End Code 0x000D

Response Timeout: 0 *100 ms (default 5000ms)
Unit Number: 0
Delay: 0 *10 ms
NT/PC Link Max: 0

● PLC Setup Settings

Programming Console address		Name	Settings
Word	Bit		
162	0 to 15	No-protocol Mode Send Delay	0000 to 210F hex, 0 to 99,990 ms decimal (in 10-ms units)
164	8 to 15	No-protocol Mode Start Code	00 to FF hex
	0 to 7	No-protocol Mode End Code	00 to FF hex
165	12	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
	8 and 9	No-protocol Mode End Code Specifier	0 hex: None 1 hex: Use end code. 2 hex: Use CR+LF.
	0 to 7	No-protocol Mode Number of Bytes of Data	00 hex: 256 bytes (default) 01 to FF hex: 1 to 255 bytes

● DM Setup Area Settings for Serial Communication Board's Ports

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
D32002	D32012	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)
D32004	D32014	8 to 15	No-protocol Mode Start Code	00 to FF hex
		0 to 7	No-protocol Mode End Code	00 to FF hex
D32005	D32015	12 to 15	No-protocol Mode Start Code Specifier	0 hex: None 1 hex: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0 hex: None 1 hex: Use end code. 2 hex: Use CR+LF.
		0 to 7	Number of Bytes of Data	00 hex: 256 bytes (default) 01 to FF hex: 1 to 255 bytes

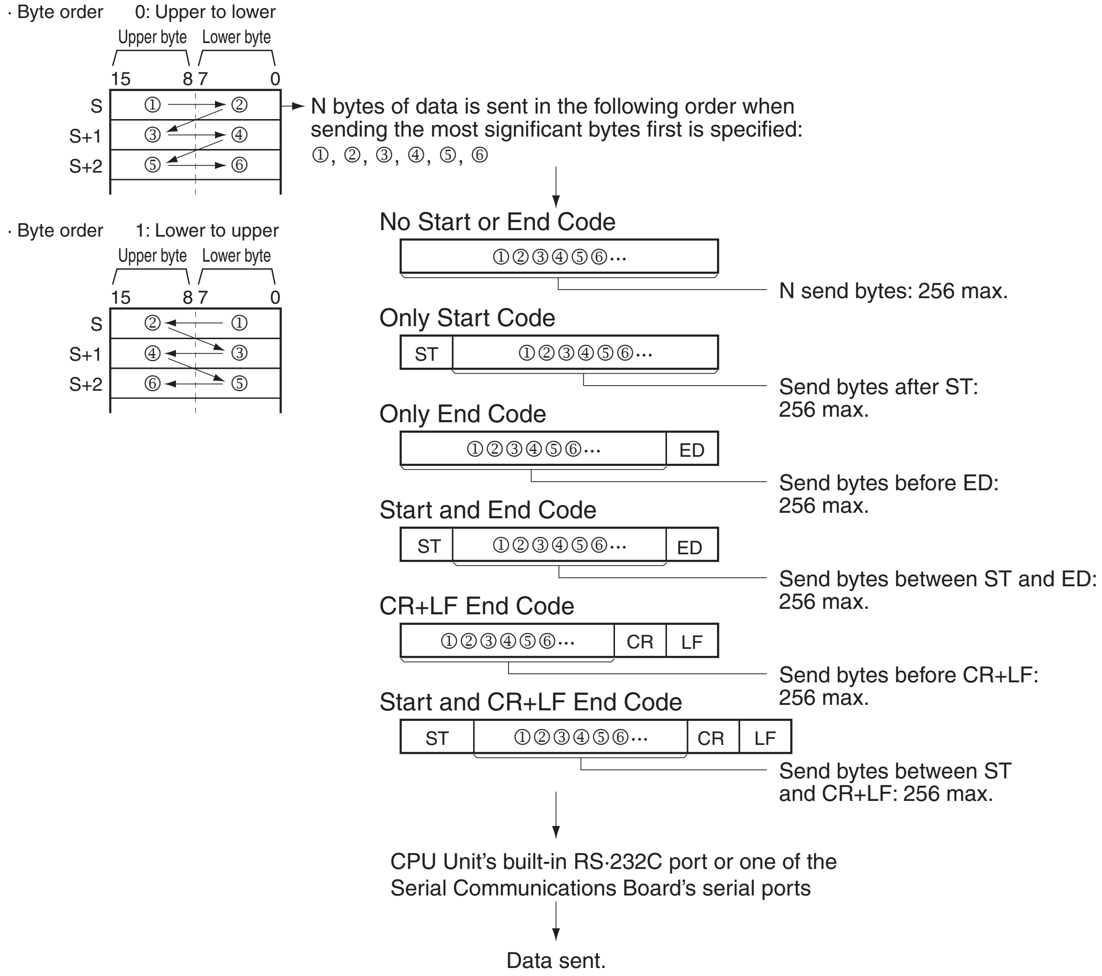
Function

- TXD(236) reads N bytes of data from words S to S+(N÷2)-1 and outputs the raw data in no-protocol mode from the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports. (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 (for the CPU Unit's RS-232C port) or the allocated DM Setup Area (for the Serial Communications Board's ports).
- The following send-message frame format can be set in the PLC Setup (for the CPU Unit's RS-232C port) or the allocated DM Setup Area (for the Serial Communications Board's ports).
  1) Start code: None or 00 to FF hex.
  2) 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 or 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 is sent in the order specified in C.
- If RS signal control is specified in C, bit 15 of S will be used as the RS signal.
- If ER signal control is specified in C, bit 15 of S will be used as the ER signal.
  If RS and ER signal control is specified in C, bit 15 of S will be used as the RS signal and bit 14 of S will be used as the ER signal.
  If 1, 2, or 3 hex is specified for RS and ER signal control in C, TXD(236) will be executed regardless of the status of the Send Ready Flag (A392.05, A356.05, or A356.13 depending on the port being used). If RS and ER signal control is set to 1, 2 or 3 hex, data will not be sent.
- Up to 259 bytes can be sent, including the send data (N = 256 bytes max.), the start code, and the end code.
- Specify the size of the send data, not including the start code and end code, in N.

Start code / end code settings and send data



Hint

- When sending data to another device by TXD instruction, the device may require that the data be sent at certain intervals. In that case, a transmission delay time can be set to adjust the transmission intervals.

Precautions

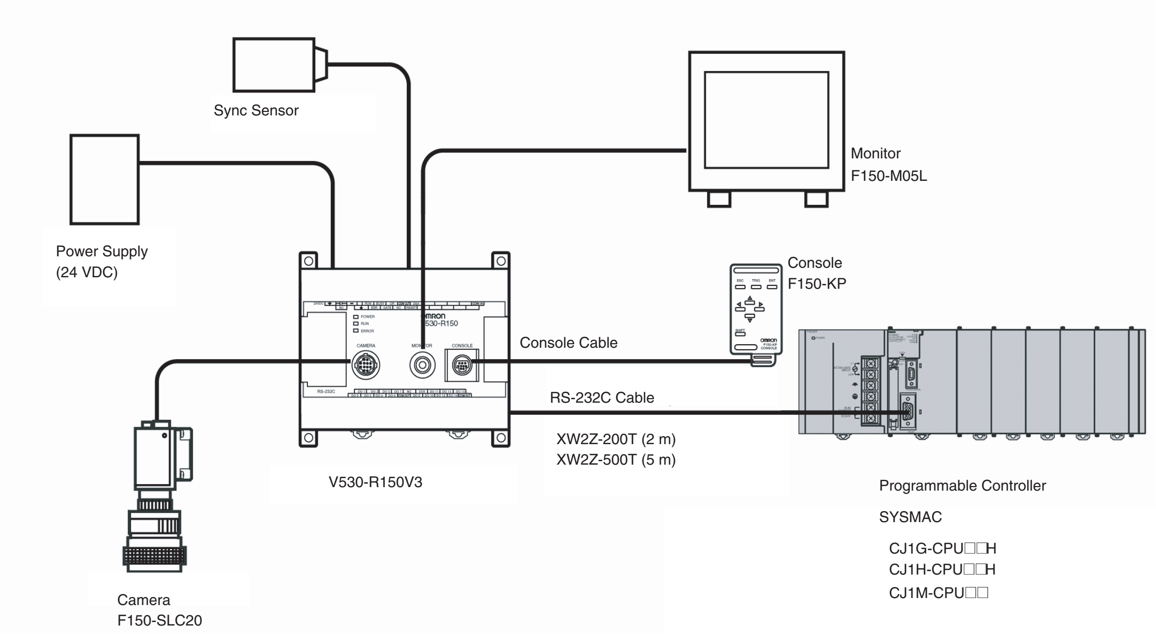
- Do not program TXD(236)/RXD(235) for a Serial Communications Board's port (port 1 or 2) in both the cyclic task and interrupt task. A TXD(236)/RXD(235) instruction cannot be executed for the Serial Communications Board in the interrupt task if a TXD(236)/RXD(235) instruction is being executed for the Serial Communications Board in the cyclic task. An error will occur and the ER Flag will be turned ON if a TXD(236)/RXD(235) instruction is executed for the Serial Communications Board in the interrupt task when another TXD(236)/RXD(235) instruction was being executed for the Serial Communications Board in the cyclic task. (These instructions cannot be programmed in both the cyclic and interrupt tasks even if they are executed for different ports in the Serial Communications Board.)
- TXD(236) can be used only for the CPU Unit's RS-232C port or one of the Serial Communications Board's serial ports. In addition, the port must be set to no-protocol mode.
- Data can be sent only when the port's Send Ready Flag is ON. (The Send Ready Flag is A392.05 for the CPU Unit's RS-232C port, A356.05 for Serial Communications Board port 1, or A356.13 for Serial Communications Board port 2.)
- Nothing will be sent if 0 is specified for N.
- If the TXD(236) instruction is executed for a Board that does not support no-protocol mode (a Serial Communications Board without a version number), the Inner Board Service Disabled Flag (A424.04) and the Error Flag will turn ON.

Example Programming

● Sending Data to a Code Reader

This example shows how to send data to the V530-R150V3 2D Code Reader as an example of communicating with an external device.

Hardware Configuration



In this example, the external device is connected to the RS-232C port built into the CPU Unit.

First, set the reading conditions for the Code Reader.

Communications Settings

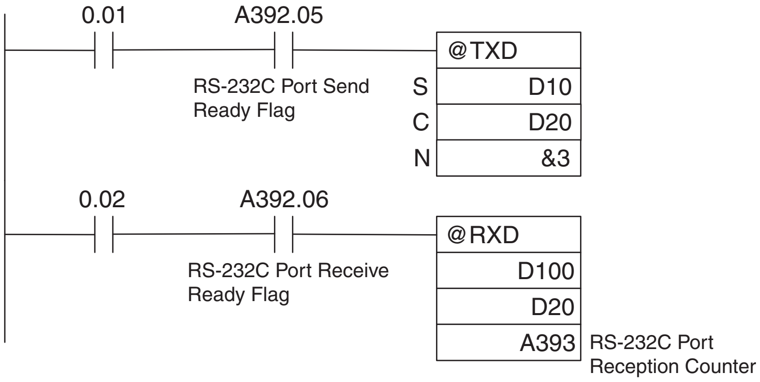
The communications settings of the Code Reader are given in the following table. These are the default settings.

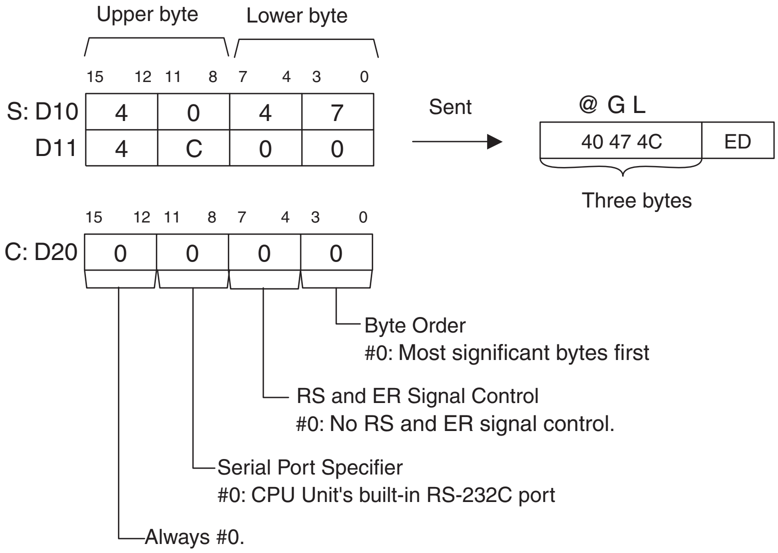
Item	Setting
Communications mode	No-protocol
Baud rate	38,400 bps
Data bit length	8 bits
Parity	None
Stop bits	1
Start code	None
End code	#000D (CR)

Set the PLC communications settings to the same values in the PLC Setup. Only the end code needs to be set.

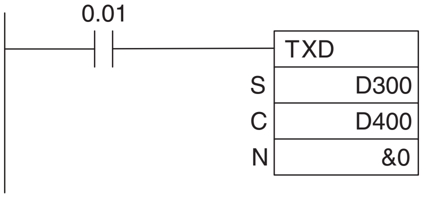
Programming Example

If CIO 0.01 turns ON while the RS-232C Port Send Ready Flag (A392.05) is ON, three bytes of data starting from the upper byte of D10 are sent without conversion to the Code Reader connected to the CPU Unit's built-in RS-232C port. These three bytes contain “@GL”, which is the normal read command used as a trigger input to the Code Reader from the RS-232C line.

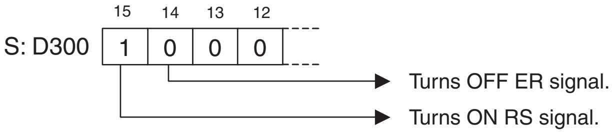


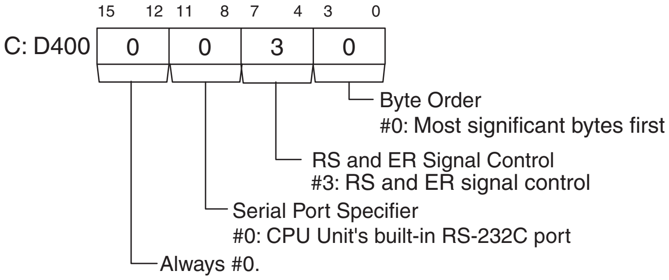


● Performing RS or ER Signal Control



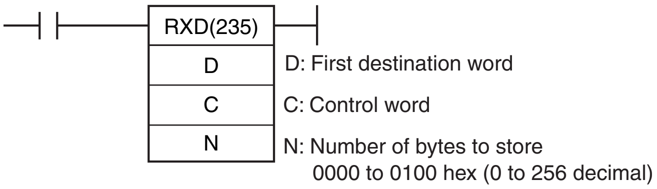
When CIO 0.01 turns ON, the status of bit 15 of D300 is output as the RS signal and the status of bit 14 is output as the ER signal.





RXD

Instruction	Mnemonic	Variations	Function code	Function
RECEIVE	RXD	@RXD	235	Reads the specified number of bytes of data from the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports. (The Serial Communications Board must be Ver. 1.2 or later).

	RXD
Symbol	

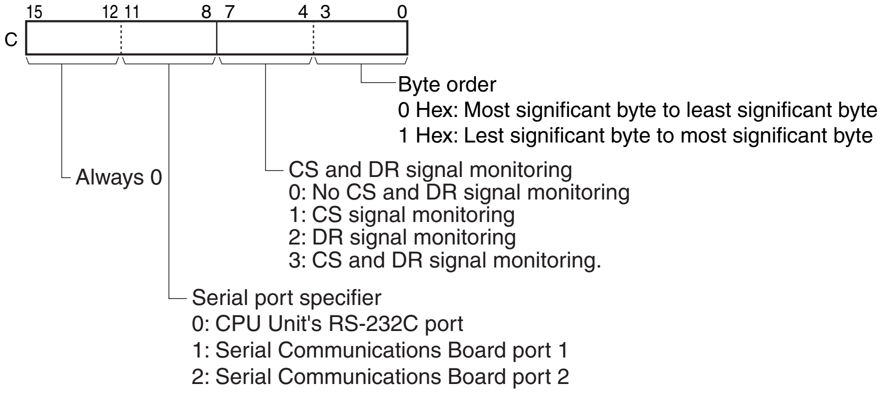
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	First destination word	UINT	Variable
C	Control word	UINT	1
N	Number of bytes to store 0000 to 0100 hex (0 to 256 decimal)	UINT	1

C: Control Word



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK	OK						
N																		

Flags

Name	Label	Operation
Error Flag	ER	• ON if the CPU Unit's RS-232C port is specified as the send port, but no-protocol mode is not set in the PLC Setup. • ON if one of the Serial Communication Board's serial ports is specified as the send port, but no-protocol mode is not set in the port's allocated DM Setup Area. ON if one of the Serial Communications Board's serial ports is specified, but the Board does not support no-protocol mode (the Board does not have a version number). • ON if the value of C is not within range. • ON if the value for N is not between 0000 and 0100 hex. • When using CS1D CPU Units, ON if active and standby CPU Units could not be synchronized. • ON (ER Flag in interrupt tasks) if a TXD(236) or RXD(235) instruction is being executed for the Serial Communications Board in the cyclic task, the cyclic task is interrupted, and another TXD(236) or RXD(235) instruction is executed for the Serial Communications Board in the interrupt task. • ON if a RXD(235) was executed for a serial port on a Serial Communications Board that was being restarted. • OFF in all other cases.

Related Auxiliary Area Words and Bits

● Auxiliary Area Flags for CPU Unit's RS-232C Port

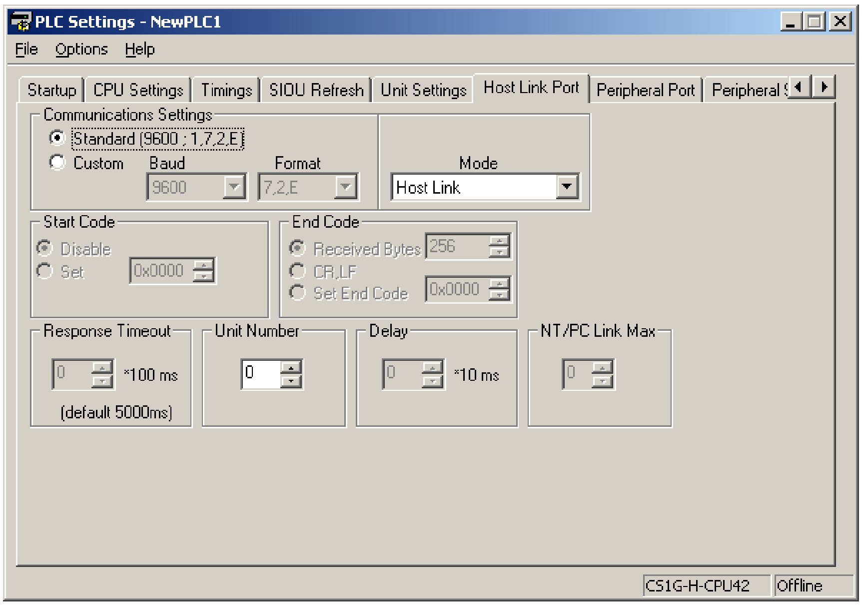
Name	Address	Contents
RS-232C Port Reception Completed Flag	A392.06	ON when no-protocol reception is completed. Number of Receive Bytes Specified: The flag will turn ON when the specified number of bytes has been received. End Code Specified: The flag will turn ON when the end code is received or when 256 bytes have been received.
RS-232C Port Reception Overflow Flag	A392.07	ON when more than the expected number of receive bytes has been received. Number of Receive Bytes Specified: The flag will turn ON when anything is received after reception has been completed and execution of the next RXD(235). End Code Specified: The flag will turn ON when anything is received after the end code has been received and execution of the next RXD(235) or when the 257th byte of data is received before the end code is received.
RS-232C Port Reception Counter	A393	Counts in hexadecimal the number of bytes received in no-protocol mode.

● Auxiliary Area Flags for Serial Communication Board's Ports

Port	Name	Address	Contents
Port 1	Reception Completed Flag	A356.06	ON when no-protocol reception is completed. Number of Receive Bytes Specified: The flag will turn ON when the specified number of bytes has been received. End Code Specified: The flag will turn ON when the end code is received or when 256 bytes have been received.
	Reception Overflow Flag	A356.07	ON when more than the expected number of receive bytes has been received in no-protocol mode. Number of Receive Bytes Specified: The flag will turn ON when more data is received after reception was completed but before the received data was not read from the buffer with RXD(235). End Code Specified: The flag will turn ON when 257 or more bytes of data are received without an end code.
	Reception Counter	A357	Counts in hexadecimal the number of bytes received in no-protocol mode (0 to 256 decimal).
	Overrun Error Flag	CIO 1908 bit 04	ON when 260 or more bytes of data are received in the buffer before RXD(235) is executed.
Port 2	Reception Completed Flag	A356.14	ON when no-protocol reception is completed. Number of Receive Bytes Specified: The flag will turn ON when the specified number of bytes has been received. End Code Specified: The flag will turn ON when the end code is received or when 256 bytes have been received.
	Reception Overflow Flag	A356.15	ON when more than the expected number of receive bytes has been received in no-protocol mode. Number of Receive Bytes Specified: The flag will turn ON when more data is received after reception was completed but before the received data was not read from the buffer with RXD(235). End Code Specified: The flag will turn ON when 257 or more bytes of data are received without an end code.
	Reception Counter	A358	Counts in hexadecimal the number of bytes received in no-protocol mode (0 to 256 decimal).
	Overrun Error Flag	CIO 1918 bit 04	ON when 260 or more bytes of data are received in the buffer before RXD(235) is executed.
Ports 1 and 2	Inner Board Service Disabled Flag	A424.04	ON when RXD(235) is executed for a Serial Communications Board that does not support no-protocol mode (a Board without a version number).

Related PLC Setup Settings for the CPU Unit's Built-in RS-232C Port

● CX-Programmer Settings



Communications Settings
● Standard (9600 ; 1,7,2,E)
○ Custom Baud 9600 Format 7,2,E
Mode Host Link

Start Code
● Disable
○ Set 0x0000

End Code
● Received Bytes 256
○ CR,LF
○ Set End Code 0x0000

Response Timeout 0 *100 ms (default 5000ms)
Unit Number 0
Delay 0 *10 ms
NT/PC Link Max 0

CS1G-H-CPU42 Offline

PLC Setup Settings for CPU Unit's RS-232C Port

Programming Console address		Name	Settings
Word	Bit		
162	0 to 15	No-protocol Mode Send Delay	0000 to 210F hex, 0 to 99,990 ms decimal (in 10-ms units)
164	8 to 15	No-protocol Mode Start Code	00 to FF hex
	0 to 7	No-protocol Mode End Code	00 to FF hex
165	12	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
	8 and 9	No-protocol Mode End Code Specifier	0 hex: None 1 hex: Use end code. 2 hex: Use CR+LF.
	0 to 7	No-protocol Mode Number of Bytes of Data	00 hex: 256 bytes (default) 01 to FF hex: 1 to 255 bytes

DM Setup Area Settings for Serial Communication Board's Ports

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
D32002	D32012	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)
D32004	D32014	8 to 15	No-protocol Mode Start Code	00 to FF hex
		0 to 7	No-protocol Mode End Code	00 to FF hex
D32005	D32015	12 to 15	No-protocol Mode Start Code Specifier	0 hex: None 1 hex: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0 hex: None 1 hex: Use end code. 2 hex: Use CR+LF.
		0 to 7	Number of Bytes of Data	00 hex: 256 bytes (default) 01 to FF hex: 1 to 255 bytes

Function

- RXD(235) reads data that has been received in no-protocol mode at the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports (the port is specified with bits 8 to 11 of C) 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.
- The following receive message frame format can be set in the PLC Setup (for the CPU Unit's RS-232C port) or the allocated DM Setup Area (for the Serial Communications Board's ports).

  1) Start code: None or 00 to FF hex

  2) 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).
- Data will be stored in memory in the order specified in C.
- Cases where the reception completion flag turns ON

  The Reception Completed Flag (note (a)) will turn ON when the number of bytes specified in the PLC Setup (for the CPU Unit's RS-232C port) or the allocated DM Setup Area (for the Serial Communications Board's ports) has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter (note (b)) 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 (note (c)) will turn ON.
  If an end code is specified in the PLC Setup or the allocated DM Setup Area, the Reception Completed Flag (note (a)) will turn ON when the end code is received or when 256 bytes of data have been received.
- Reception will be stopped if 259 bytes of data are received. If more data is input after that, the Overrun Error Flag (note (e)) and Transmission Error Flag (note (f)) will turn ON. If an overrun error, framing error, or parity error occurs on the CPU Unit's built-in serial port, serial port reception will stop. The serial port must be restarted to begin reception again.

- When more data is input to the Serial Communications Board's serial port than is specified in N, that data will be discarded when RXD(235) is executed. In contrast, extra data input to the CPU Unit's RS-232C port will not be discarded when RXD(235) is executed.
- When RXD(235) is executed, data is stored in memory starting at D, the Reception Completed Flag (note (a)) will turn OFF (even if the Reception Overflow Flag (note (c)) is ON).
- With the CPU Unit's built-in RS-232C port, if the RS-232C Port Restart Bit (note (d)) is turned ON, the Reception Completed Flag (note (a)) will be turned OFF (even if the Reception Overflow Flag is ON), and the Reception Counter (note (b)) will be cleared to 0.
- Specification of monitor in bits 4 to 7 of C for the CS and DR signals takes effect as follows:
    1) If CS signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D.
    2) If DR signal monitoring is specified in C, the status of the DR signal will be stored in bit 15 of D.
    3) 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.
- If 1, 2, or 3 hex is specified for CS and DR signal control in C, RXD(235) will be executed regardless of the status of the Receive Completed Flag (note (a)).
- Receive data will not be stored if CS or DR signal monitoring is specified.
- Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.
- Specify the size of the receive data, not including the start code and end code, in N.

Note Related Auxiliary Area and CIO Area Addresses

(a) Reception Completed Flags

Built-in RS232C port	A392.06
Serial Communications Board port 1:	A356.06
Serial Communications Board port 2:	A356.14

(b) Reception Counters

Built-in RS232C port	A393
Serial Communications Board port 1:	A357
Serial Communications Board port 2:	A358

(c) Reception Overflow Flags

Built-in RS232C port	A392.07
Serial Communications Board port 1:	A356.07
Serial Communications Board port 2:	A356.15

(d) RS-232C Port Restart Bit

Built-in RS232C port	A526.00

(e) Overrun Error Flags

Serial Communications Board port 1:	CIO 1908.04
Serial Communications Board port 2:	CIO 1918.04

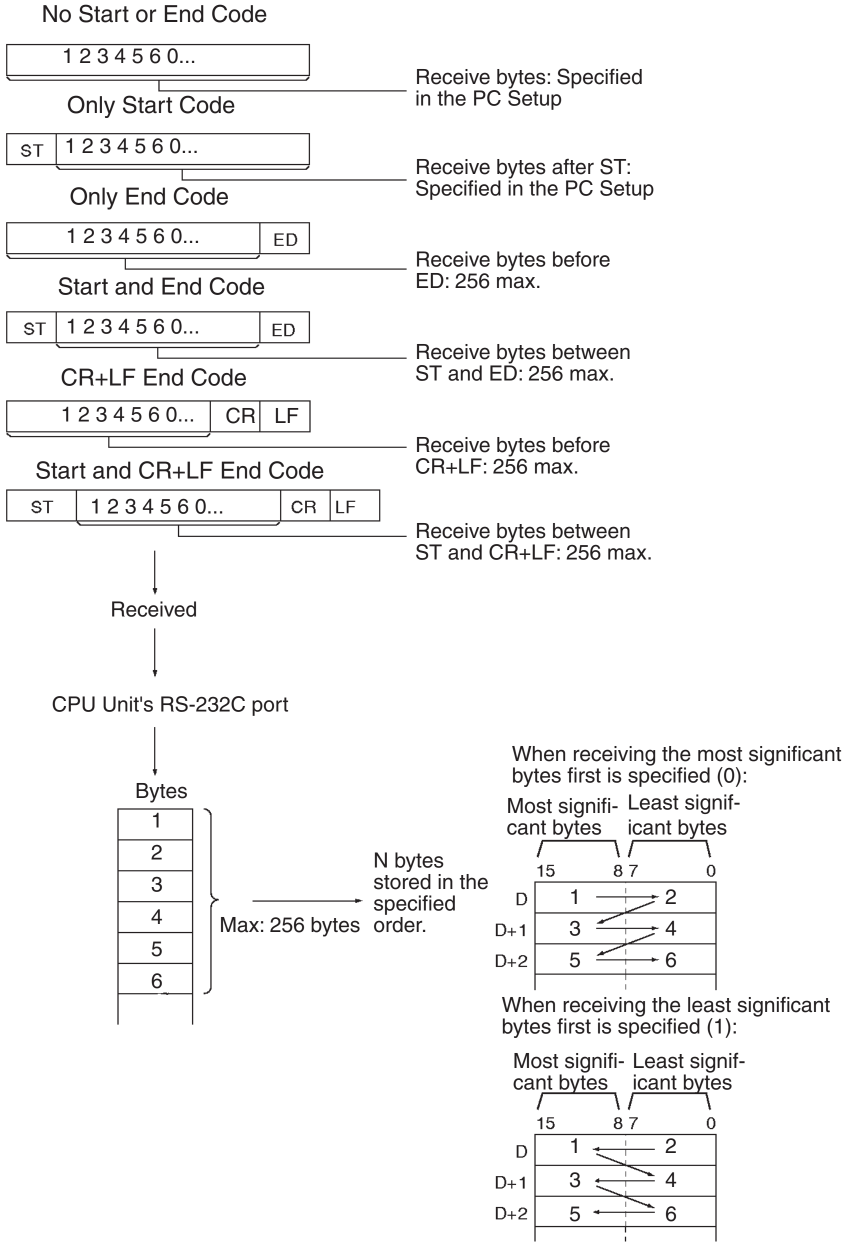
(f) Transmission Error Flags

Serial Communications Board port 1:	CIO 1908.15
Serial Communications Board port 2:	CIO 1918.15

(g) Inner Board Service Disabled Flag

Serial Communications Board ports 1 and 2:	A424.04

Start Code/End Code Settings and Receive Data

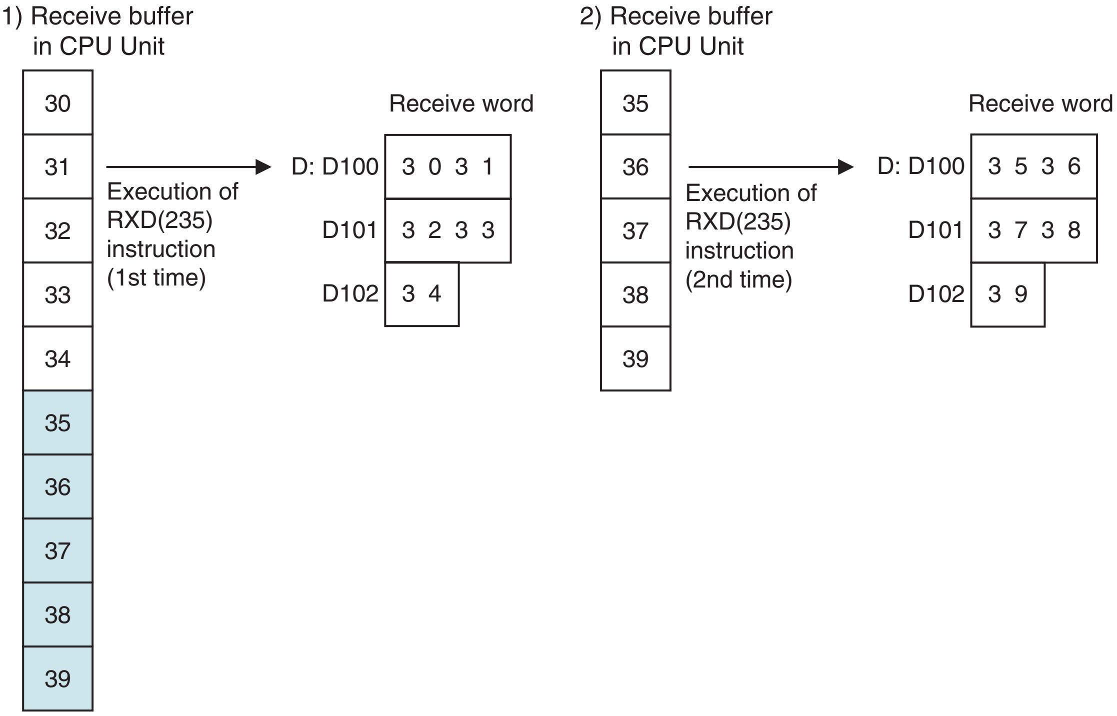


Hint

- When the RXD(235) instruction is used to read data in the CPU Unit's RS-232C port, the data after the stored number of bytes in the reception buffer is not cleared after the command is executed, so multiple RXD(235) instructions can be executed to receive a block of data in parts.
  By contrast, when the RXD(235) instruction is used to read data in one of the Serial Communications Board's ports (Serial Communications Board Version 1.2 or later), all data in the port's reception buffer is cleared. Consequently, RXD(235) cannot be executed repeatedly to read a block of data in parts.
  An example of data reception in parts (CPU Unit) is given below.
  - Message on transmission path

30	31	32	33	34	35	36	37	38	39

In the example below, No-protocol Mode Start Code Specifier is set to None, and Number of Bytes of Data is set to 10 bytes. When the RXD(235) instruction is executed with the number of bytes to store set to &5 (byte order is upper byte to lower byte), the receive buffer and receive word are as shown below.



Precautions

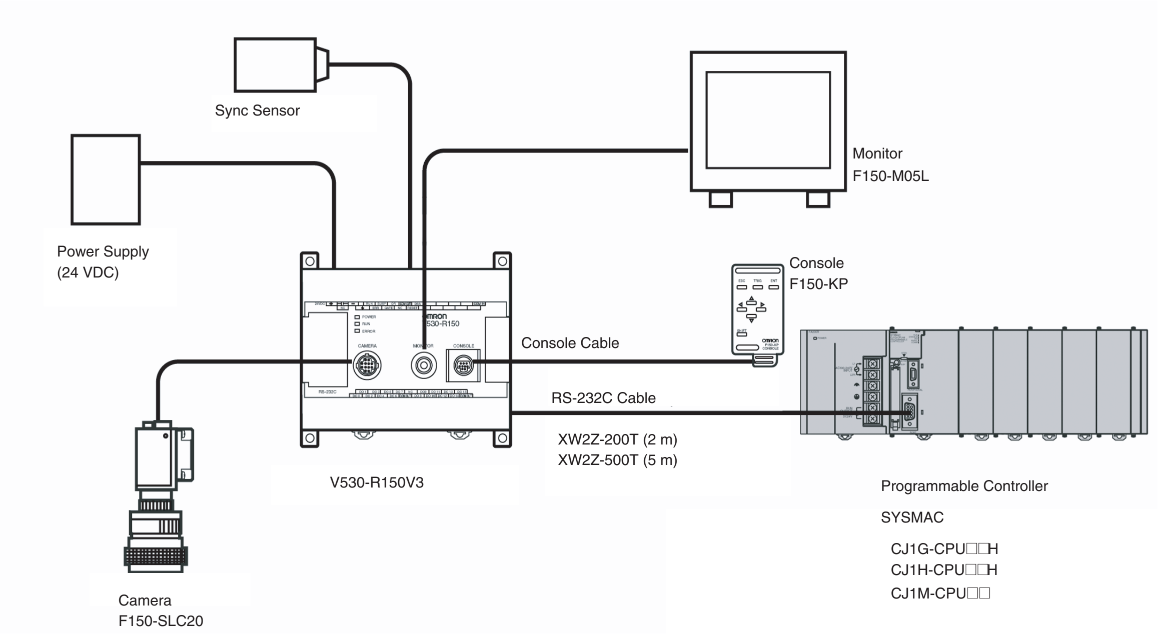
- Do not enter a TXD instruction or RXD instruction for a serial port (port 1 or 2) of the serial communication board in both a cyclic task and an interrupt task. If an interrupt task is started during the execution of a TXD instruction or RXD instruction for a serial port (port 1 or 2) of the serial communication board in a cyclic task, the TXD instruction or RXD instruction for a serial port (port 1 or 2) of the serial communication board cannot be executed in the interrupt task. An error will occur and the error flag immediately after the TXD or RXD instruction in the interrupt task will turn ON. (No combination of TXD and RXD instructions and ports 1 and 2 in a cyclic task and interrupt task is possible.)
- RXD(235) can be used only for the CPU Unit's RS-232C port or one of the Serial Communications Board's serial ports. In addition, the port must be set to no-protocol mode.
- Execute this instruction when the reception completion flag (RS-232C incorporated in the CPU Unit: A392.06, serial communication port 1: A356.06, or port 2: A356.14) is 1 (ON) to receive data (from the reception buffer).
- When data is received, the data must be read by an RXD instruction or the next data cannot be received. When the reception completion flag turns ON, read the received data with an RXD instruction before the next reception
- Specify the size of the receive data, not including the start code and end code, in N.
- 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 the RXD(235) instruction is executed for a Board that does not support no-protocol mode (a Serial Communications Board without a version number), the Inner Board Service Disabled Flag (A424.04, non-fatal error) and the Error Flag will turn ON.

Example Programming

● Receiving data

This example shows how to receive data from the V530-R150V3 2D Code Reader as an example of communicating with an external device.

Hardware Configuration



In this example, the external device is connected to the RS-232C port built into the CPU Unit.

First, set the reading conditions for the Code Reader.

Communications Settings

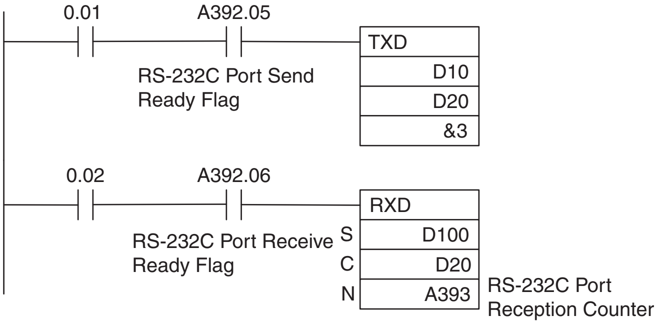
The communications settings of the Code Reader are given in the following table. These are the default settings.

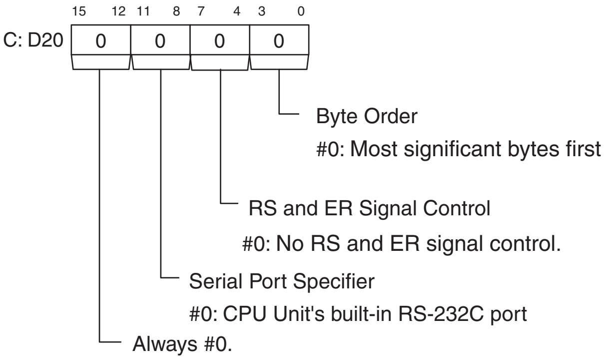
Item	Setting
Communications mode	No-protocol
Baud rate	38,400 bps
Data bit length	8 bits
Parity	None
Stop bits	1
Start code	None
End code	#000D (CR)

Set the PLC communications settings to the same values in the PLC Setup. Only the end code needs to be set.

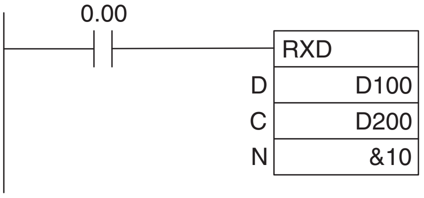
Programming Example

If CIO 0.02 turns ON while the RS-232C Port Send Ready Flag (A392.05) is ON, the number of bytes of reading results specified in the RS-232C Port Reception Counter (A393) are read from the Code Reader connected to the CPU Unit's built-in RS-232C port and stored starting from the upper byte of D100.

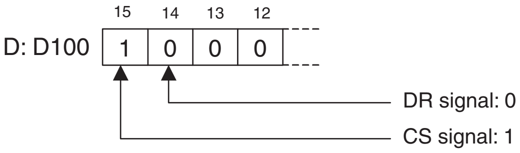


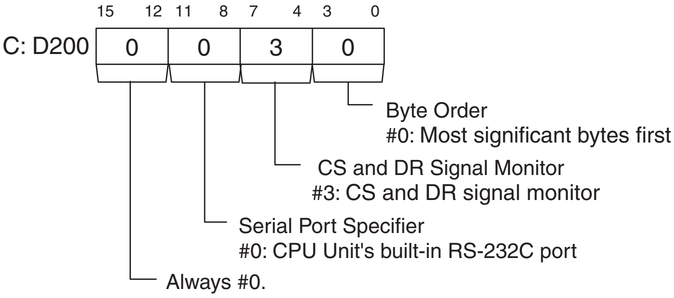


Controlling Signals



When CIO 0.00 turns ON, the status of the CS signal is output to bit 15 of D100 and the status of the DR signal is output to bit 14 of D100.

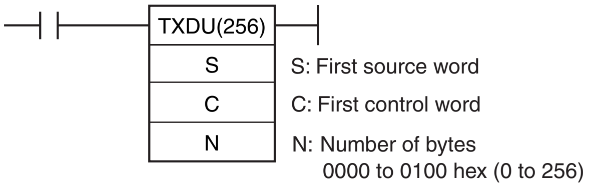




TXDU

Instruction	Mnemonic	Variations	Function code	Function
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	@TXDU	256	Outputs the specified number of bytes of data from one of the Serial Communications Unit's serial ports. (The Serial Communications Unit must be Ver. 1.2 or later).

Symbol — **TXDU**



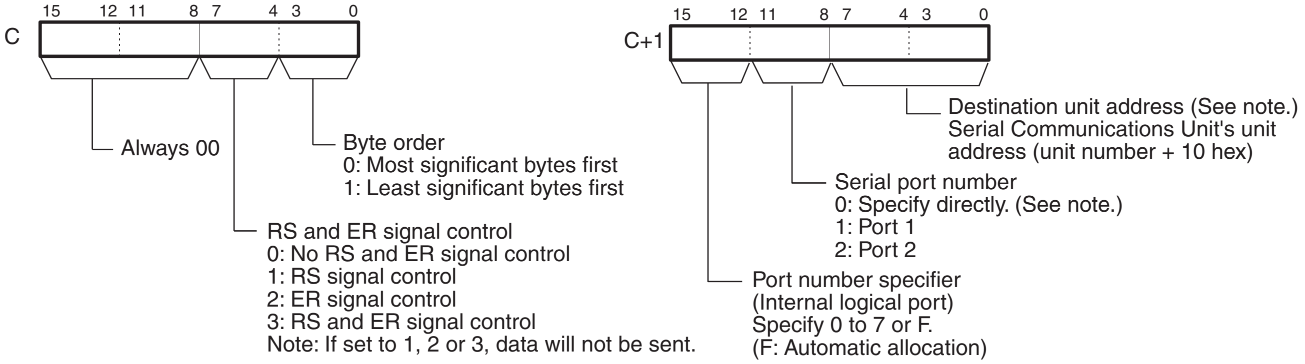
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	UINT	Variable
C	Control word	UDINT	2
N	Number of bytes 0000 to 0100 hex (0 to 256)	UINT	1

C: Control word



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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

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. • ON if TXDU(256) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

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
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.
		0 to 7	Size of receive data	#00 hex (default value): 256 bytes
				#01 to FF hex: 1 to 256 bytes

Related Auxiliary Area Words and Bits

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.

Function

- TXDU(256) reads N bytes of data from words S to S+(N+2)-1 and outputs the raw data in no-protocol mode to 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 following send-message frame formats can be set in the allocated DM Setup Area.

  1) Start code: None or 00 to FF hex.

  2) 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 is sent in the order specified in C.
- The control specification for RS and ER signals specified in bits 4 to 7 of C takes effect as shown below.

  1) If RS signal control is specified in C, bit 15 of S will be used as the RS signal.

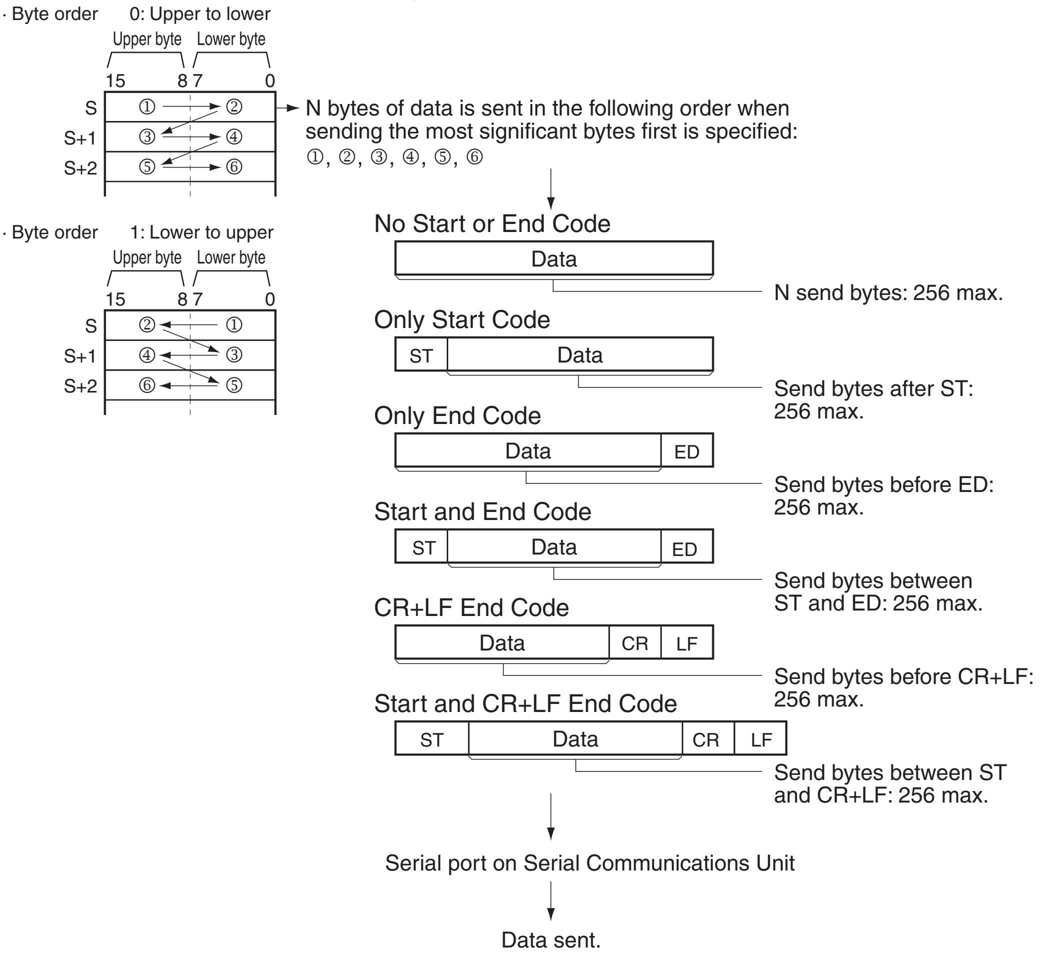
  2) If ER signal control is specified in C, bit 15 of S will be used as the ER signal.

  3) 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.

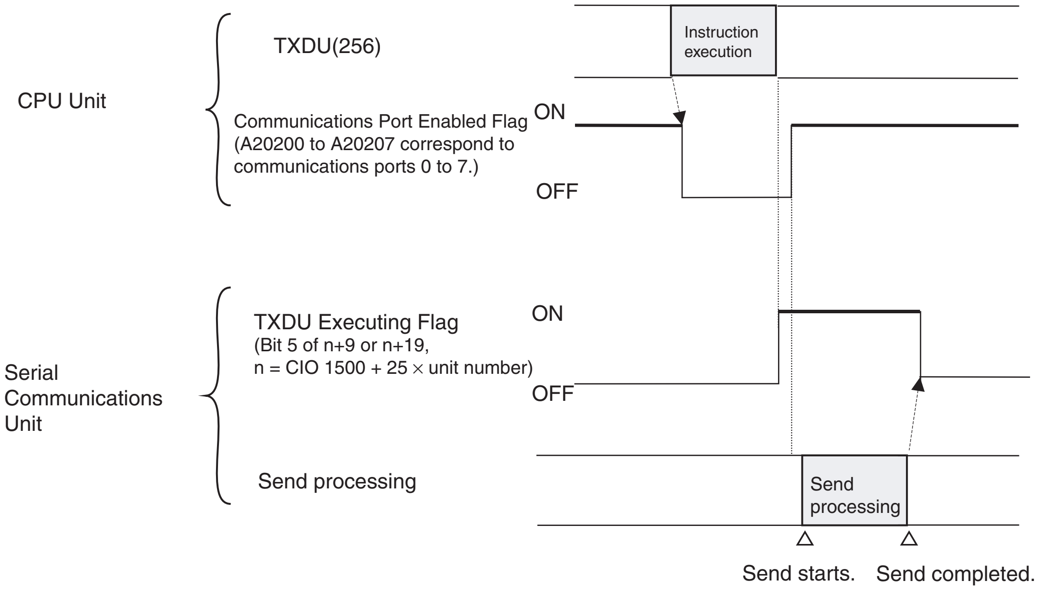
  If RS and ER signal control is set to 1, 2 or 3 hex, data will not be sent.

- The maximum number of bytes that can be sent is 259 (send data: maximum of 256 bytes, start code: 1 byte, end code: CR+LF specification 2 bytes).
- Specify the size of the send data, not including the start code and end code, in N.
- The logical port number can be allocated automatically by setting bits 12 to 15 of C+1 to F. For details, refer to *Automatically Allocating Communications Ports (i.e., Internal Logical Ports)* in *SECTION 3 Instructions* - *Network Instructions*.

Start code / end code settings and send data



Flag operation

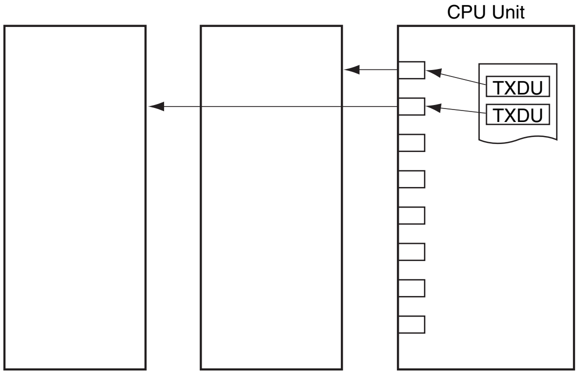


Hint

- Depending on the external device, it might be necessary to set a send delay when sending data with TXDU(256). If a send delay is required, set or adjust the delay time in the allocated DM Setup Area.

Precautions

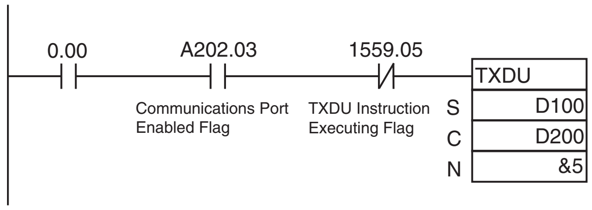
- TXDU(256) can be used only for a Serial Communications Unit's serial port that has been set to no-protocol mode.
- Transmission is only possible when the Communications Port Enabled Flag corresponding to the communication port number that is used (A202.00 to 202.07) is 1 (ON) and the TXDU Instruction Executing Flag (allocated DM setup area) is 0 (OFF).)
- Nothing will be sent if 0 is specified for N.
- 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.

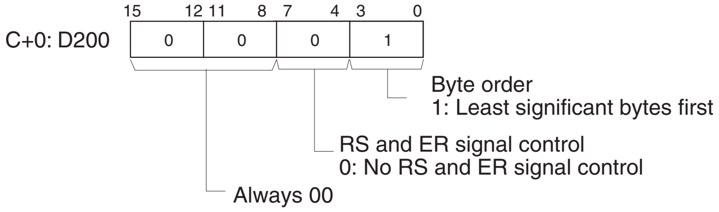


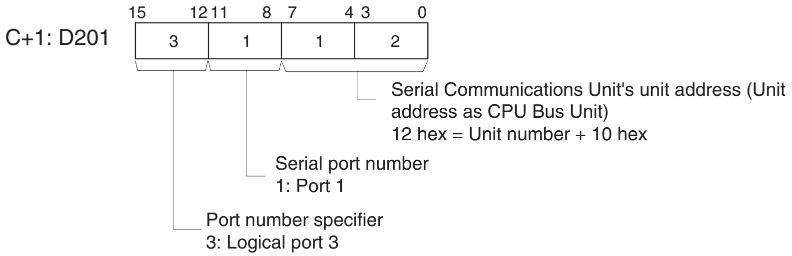
- TXDU(256) can not be executed while the TXDU Instruction Executing Flag (bit 5 of n+9 or n+19, where n = CIO 1500 + 25 × unit number) is ON. To ensure that another TXDU(256) is not executed for the port before the first TXDU(256) is completed, program the port's TXDU Instruction Executing Flag as a normally closed condition.

Example Programming

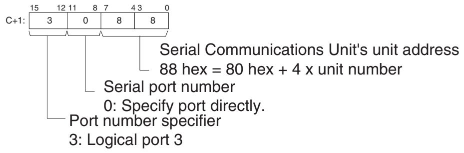
When CIO 0.00 is ON, A202.03 (the Communications Port Enabled Flag) is ON, and CIO 1559.05 (the TXDU Instruction Executing Flag for port 1) is OFF in the following example, TXDU(256) outputs data through serial port 1 of the Serial Communications Unit with unit number 2. The 5 bytes of output data are 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.

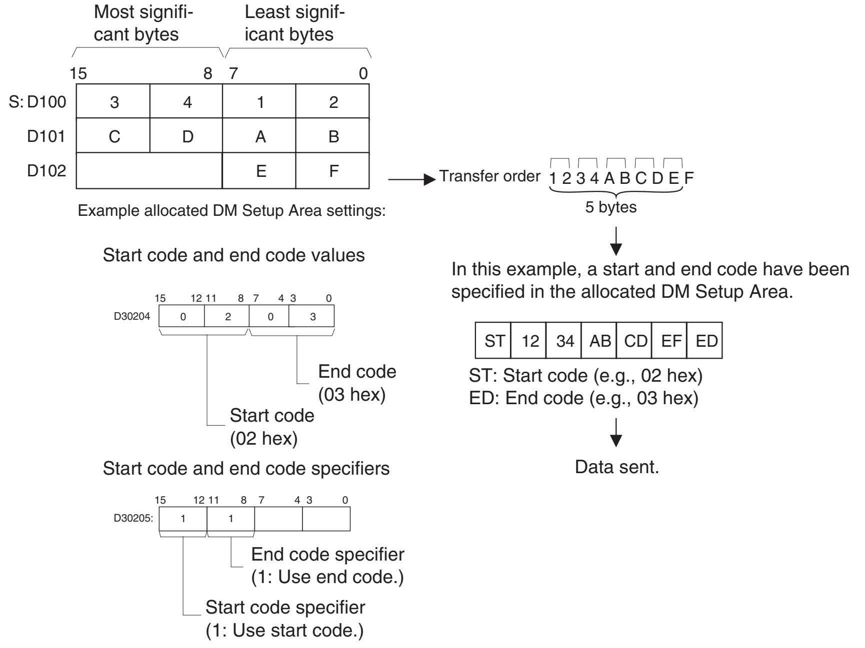




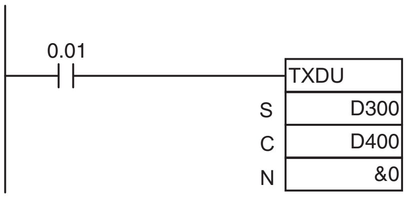


Note:
The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the Serial Communications Unit's unit address to the serial port's unit address.
(Set the unit address to 80 hex + 4 × unit number for port 1 or 81 hex + 4 × unit number for port 2.)

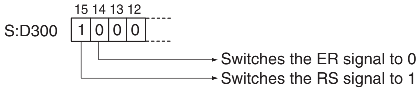


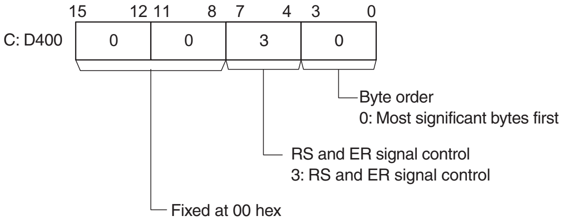


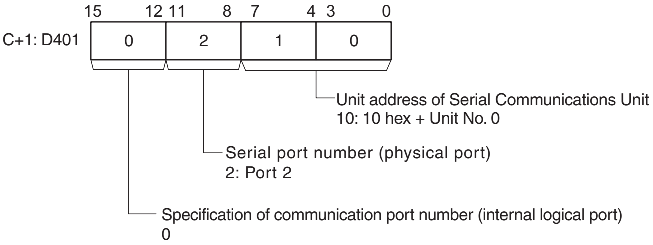
● Performing RS and ER signal control



When 0.01 is ON, bit 15 of D300 is applied to the RS signal, and bit 14 will be used as the ER signal.

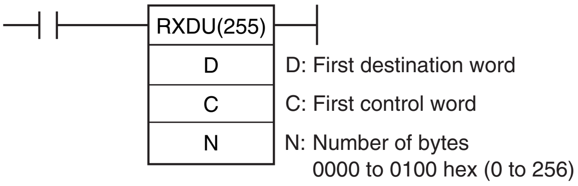






RXDU

Instruction	Mnemonic	Variations	Function code	Function
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	@RXDU	255	Reads the specified number of bytes of data from one of the Serial Communications Unit's serial ports. (The Serial Communications Unit must be Ver. 1.2 or later).

	RXDU
Symbol	

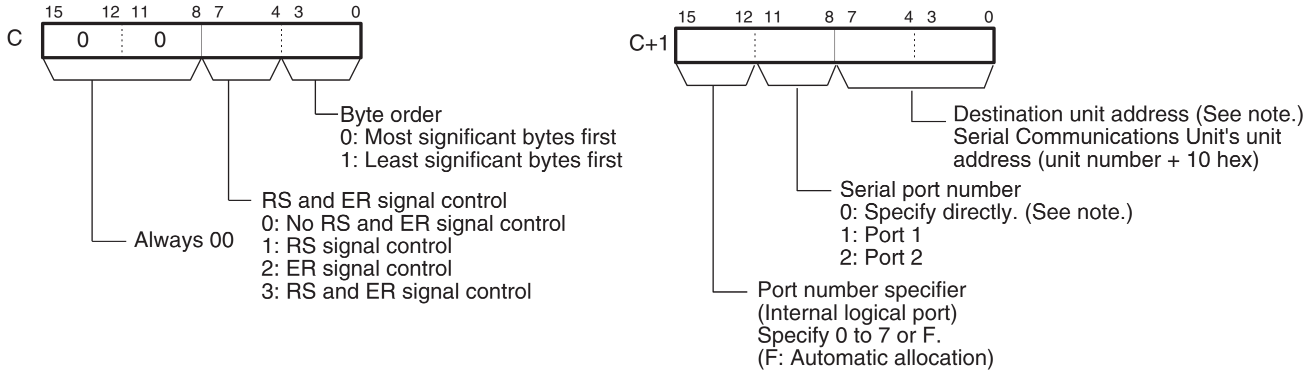
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	First destination word	UINT	Variable
C	First control word	UDINT	2
N	Number of bytes 0000 to 0100 hex (0 to 256)	UINT	1

C: Control word



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	Word addresses								Indirect DM/EM addresses		Con- stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	• ON if all of the logical ports are being used or the Communications Port Enabled Flag (A202.00 to A202.07) 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. • ON if RXDU(255) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

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.
		0 to 7	Size of receive data	#00 hex (default value): 256 bytes #01 to FF hex: 1 to 256 bytes

Related Auxiliary Area Words and Bits

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.
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	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:RXD(235) or 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 Board or Unit has not received more than the specified number of bytes. 1: The Board or Unit has received more than the specified number of bytes. 0 → 1:The Board or Unit received more data after data reception was completed. 1 → 0:RXD(235) or RXDU(255) was executed to write the data from the buffer to a CPU Unit data area.
n+10	n+20	00 to 15	Reception Counter Indicates the number of bytes received in hexadecimal, between 0000 and 0100 hex (0 to 256 decimal).

Function

- 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 following receive-message frame formats can be set in the allocated DM Setup Area.
  1) Start code: None or 00 to FF hex.
  2) 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).
- Data will be stored in memory in the order specified in C.
- Cases where the Reception Completion Flag (*1) turns ON
  1) The Reception Completed Flag (note (a)) will turn ON when the number of bytes specified in the allocated DM Setup Area has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter (note (b)) will have the same value as the number of receive bytes specified in the allocated DM Setup Area.
  2) If an end code is specified in the allocated DM Setup Area, the Reception Completed Flag (note (a)) will turn ON when the end code is received or when 256 bytes of data have been received.
- Cases where the Reception Completion Flag (*1) turns ON
  1) If more data is received before RXDU(255) is executed after the Reception Completed Flag (note (a)) turns ON, the Reception Overflow Flag (note (c)) will turn ON.
  2) If more bytes than specified in the allocated DM Setup Area are received than specified, the Reception Overflow Flag (note (c)) 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 (note (d)) and Transmission Error Flag (note (e)) will turn ON.
- When more data is input to the Serial Communications 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 (note (a)) will turn OFF (even if the Reception Overflow Flag (note (c)) is ON), and the Reception Counter (note (b)) will be cleared to 0.
- Specification of monitor in bits 4 to 7 of C for the CS and DR signals takes effect as follows:
  1) If CS signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D.
  2) If DR signal monitoring is specified in C, the status of the DR signal will be stored in bit 15 of D.

3) 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.

- 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 (note (a)).
- Receive data will not be stored if CS or DR signal monitoring is specified.
- Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.
- Specify the size of the receive data, not including the start code and end code, in N.
- 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 logical port number can be allocated automatically by setting bits 12 to 15 of C+1 to F. For details, refer to *Automatically Allocating Communications Ports (i.e., Internal Logical Ports)* in *SECTION 3 Instructions* - *Network Instructions*.

Note Related Auxiliary Area and CIO Area Addresses

(a) Reception Completed Flags (n = CIO 1500 + 25 × unit number)

Port 1: Bit 6 of n+9

Port 2: Bit 6 of n+19

(b) Reception Counters (n = CIO 1500 + 25 × unit number)

Port 1: n+10

Port 2: n+20

(c) Reception Overflow Flags (n = CIO 1500 + 25 × unit number)

Port 1: Bit 7 of n+9

Port 2: Bit 7 of n+19

(d) Overrun Error Flags (n = CIO 1500 + 25 × unit number)

Port 1: Bit 4 of n+8

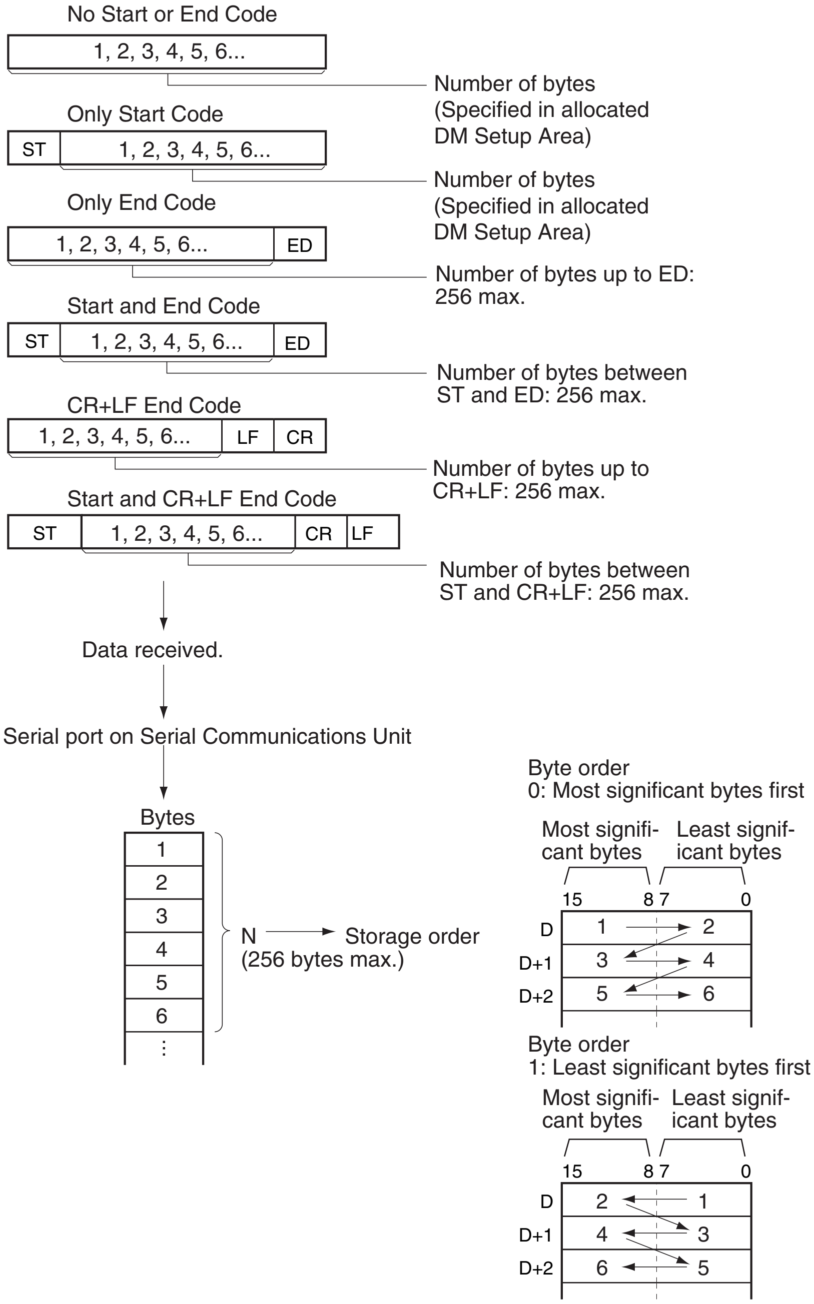
Port 2: Bit 4 of n+18

(e) Transmission Error Flags (n = CIO 1500 + 25 × unit number)

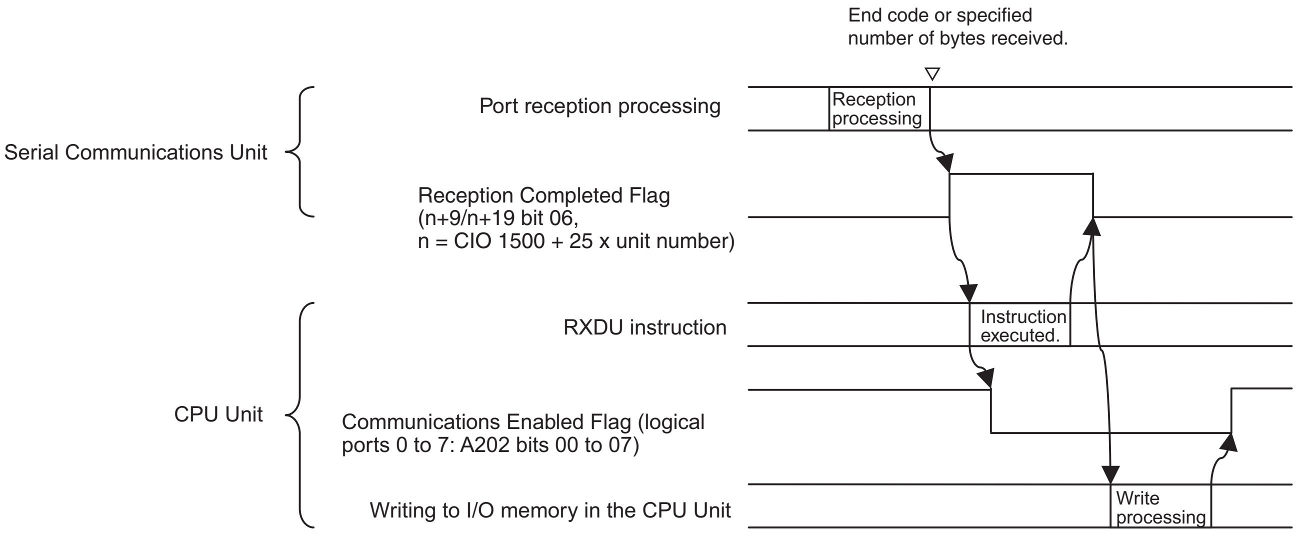
Port 1: Bit 15 of n+8

Port 2: Bit 15 of n+18

● Start Code/End Code Settings and Send Data



● Flag Operation

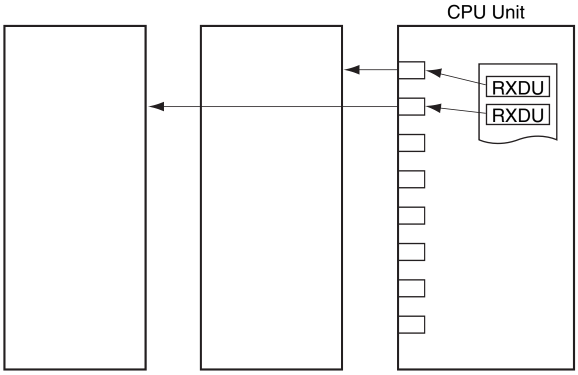


Hint

- Execute RXDU(255) to read the received data from the buffer when the Reception Completed Flag (in the allocated DM Setup Area) is ON.

Precautions

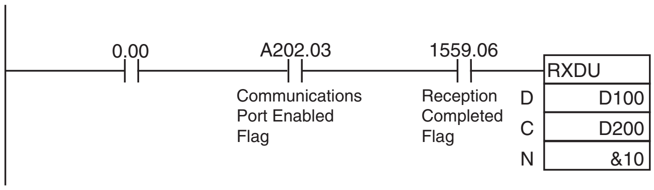
- Further data cannot be received until the received data is read from the buffer with RXDU(255). When the Reception Completed Flag goes ON, read that data promptly with RXDU(255) before more data is input to the port.
- When RXDU(255) is used to read data that was received at one of the Serial Communications Unit's ports, the port's reception buffer is cleared after RXDU(255) is executed. Consequently, RXDU(255) can not be executed repeatedly to read a block of data in parts.
- RXDU(255) can be used only for a Serial Communications Unit's serial port that has been set to no-protocol mode.
- 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.
- RXDU(255) uses a logical port (because it sends an internal FINS command) to output a receive sequence command to a Serial Communications Unit or CS-series Serial Communications Board. 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.

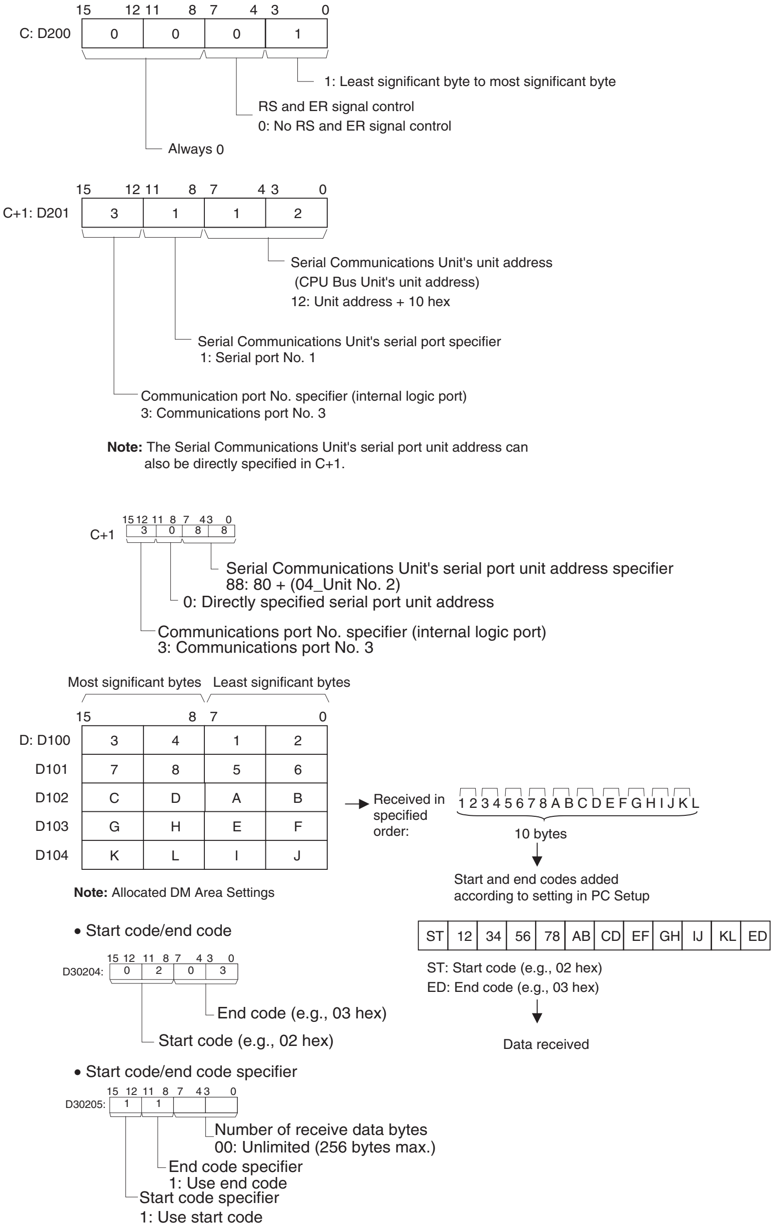


- RXDU(255) can not be executed when the Reception Completed Flag (bit 6 of n+9 or n+19, where n = CIO 1500 + 25 × unit number) is OFF. Program the Reception Completed Flag as a normally open condition of RXDU(255).

Example Programming

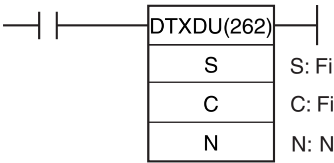
When CIO 0.00 is ON, A202.03 (the Communications Port Enabled Flag) is ON, and CIO 1559.06 (the Reception Completed Flag for port 1) is ON in the following example, RXDU(255) reads the data received through serial port 1 of the Serial Communications Unit with unit number 2. (Logical communications port 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.





DTXDU

Instruction	Mnemonic	Variations	Function code	Function
DIRECT TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	DTXDU	@DTXDU	262	Outputs the specified number of bytes of data from the serial port of a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit. The data is output in no-protocol mode from the specified first word with the start code and end code (if any) specified in the allocated DM Setup Area. This instruction sends the data to the Serial Communications Unit as soon as the instruction is executed to achieve high-speed data transmission.

	DTXDU
Symbol	 S: First source word C: First control word N: Number of bytes 0000 to 0100 hex (0 to 256)

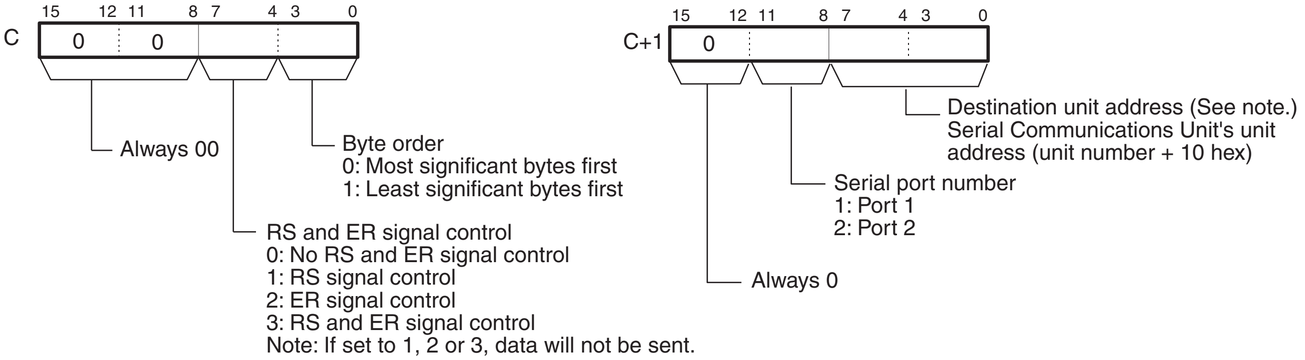
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	UINT	Variable
C	Control word	UDINT	2
N	Number of bytes 0000 to 0100 hex (0 to 256)	UINT	1

C: Control word



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C											OK							
N												OK						

Flags

Name	Label	Operation
Error Flag	P_ER	• 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 execution of DRXDU(261)/DTXDU(262) in a cyclic task is interrupted by an interrupt task and DTXDU(262) is also executed in the interrupt task. • ON if the specified Unit is not a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit. (In this case, up to 1 ms will be required to execute the instruction.) • ON if the specified Serial Communications Unit is being initialized. • ON if the specified port on the Serial Communications Unit is not in No-protocol Mode. • ON if DTXDU(262) is executed when the DTXDU Send Ready Flag is OFF. • OFF in all other cases.

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
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.
		0 to 7	Size of Data	#00 hex (default value): 256 bytes
				#01 to FF hex: 1 to 256 bytes

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	04	DTXDU Send Ready Flag	ON: Ready to send, OFF: Sending not possible

Function

- DTXDU(262) reads N bytes of data from words S to S+(N+2)-1 and outputs the raw data in no-protocol mode to the CJ1W-SCU22/SCU32/SCU42 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.
- This instruction sends the specified data to the Serial Communications Unit as soon as the instruction is executed to achieve high-speed data transmission.
- The following send-message frame formats can be set in the allocated DM Setup Area.

  1) Start code: None or 00 to FF hex.

  2) 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.

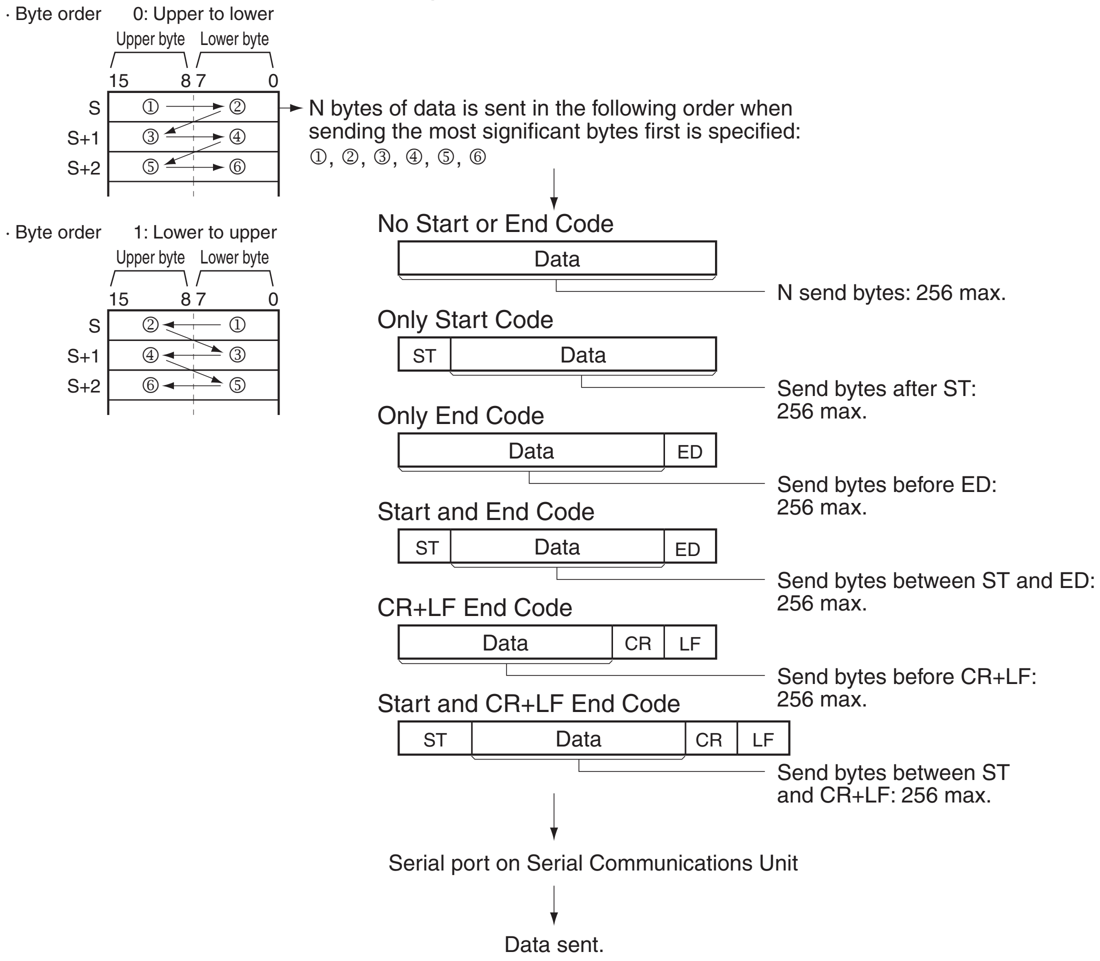
- Data is sent in the order specified in C.
- The control specification for RS and ER signals specified in bits 4 to 7 of C takes effect as shown below.

  1) If RS signal control is specified in C, bit 15 of S will be used as the RS signal.

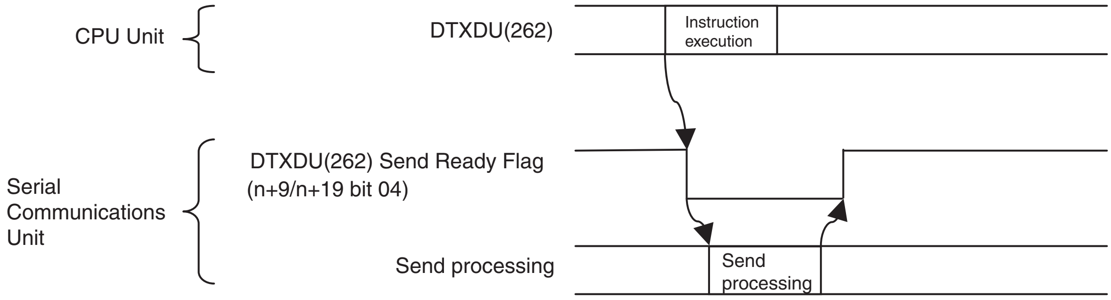
  2) If ER signal control is specified in C, bit 15 of S will be used as the ER signal.

  3) 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.
- The maximum number of bytes that can be sent is 259 (send data: maximum of 256 bytes, start code: 1 byte, end code: CR+LF specification 2 bytes). If RS and ER signal control is set to 1, 2 or 3 hex, data will not be sent.
- Specify the size of the send data, not including the start code and end code, in N.

● Start code / end code settings and send data



● Flag operation



Note Sending processing will always be started immediately when DTXDU(262) is executed.

Hint

- Depending on the external device, it might be necessary to set a send delay when sending data with DTXDU(262). If a send delay is required, set or adjust the delay time in the allocated DM Setup Area.

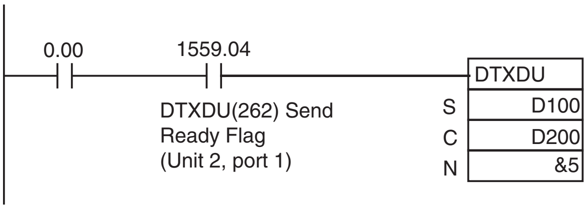
Precautions

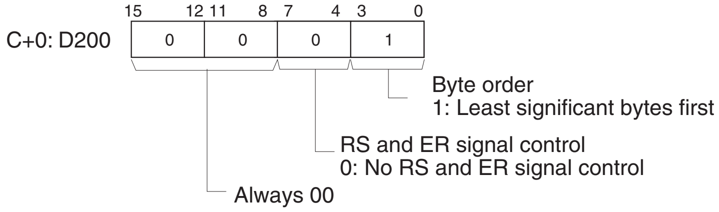
- DTXDU(262) can be executed only for serial ports on a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit. If it is executed for any other CPU Bus Unit, approximately 1 ms will be required to execute the instruction, increasing the cycle time.
- DTXDU(262) can be used only for serial ports set to No-protocol Mode on a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit mounted to a CJ2H CPU Unit with unit version 1.1 or later.
- Nothing will be sent if 0 is specified for N.

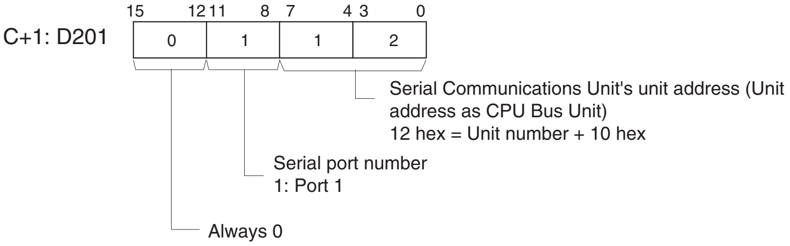
- DTXDU(262) will not be executed while the DTXDU(262) Send Ready Flag (n+9/n+19 bit 04) is OFF (n = CIO 1500 + 25 × unit number). Use the DTXDU(262) Send Ready Flag in an NO input condition for DTXDU(262).

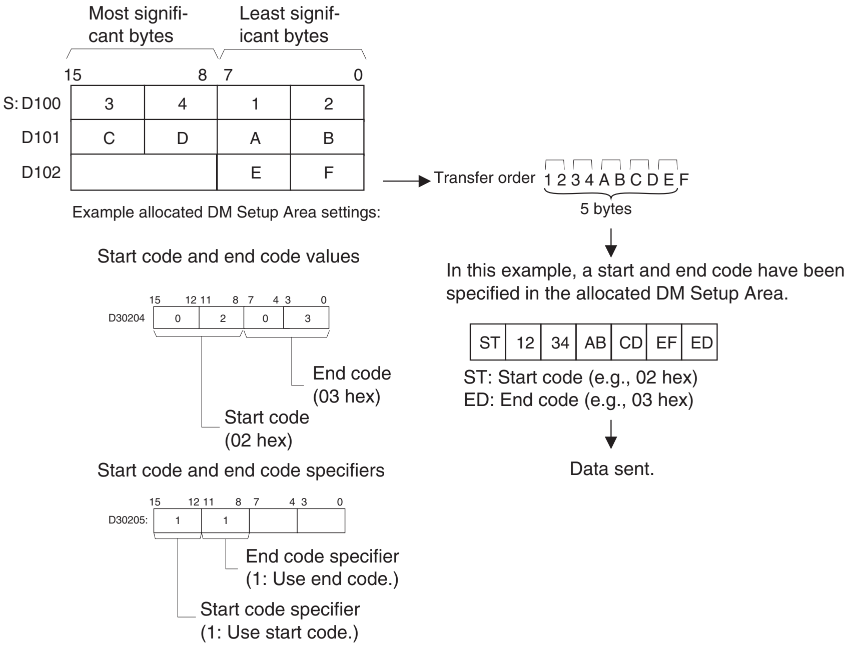
Example Programming

In the following sample, if CIO 0.00 is turned ON while the DTXDU(262) Send Ready Flag (CIO 1559.04) is ON, 5 bytes of data starting from the lower byte of D100 will be sent to the external device connected to port 1 of the Serial Communications Unit with unit number 2 without converting the data in any way.

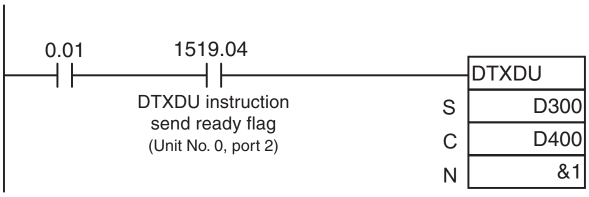




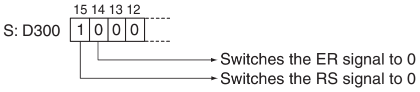


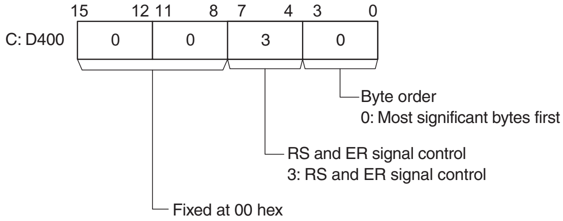


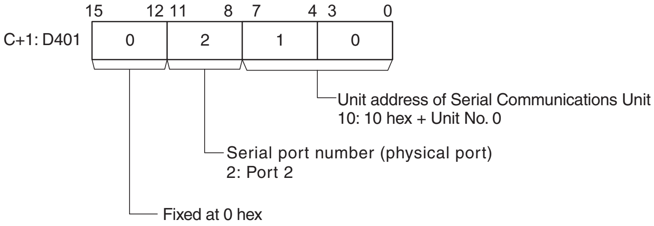
● Performing RS and ER signal control



When 0.01 is ON, bit 15 of D300 is applied to the RS signal,
and bit 14 is applied to the ER signal.





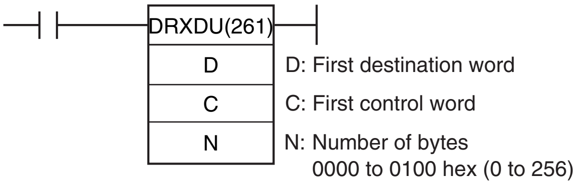


DRXDU

Instruction	Mnemonic	Variations	Function code	Function
DIRECT RECEIVE VIA SERIAL COMMUNICATIONS UNIT	DRXDU	@DRXDU	261	Reads the specified number of bytes of data from the serial port of a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit to CPU Unit memory starting at the specified first word. The data is read in no-protocol mode with the start code and end code (if any) specified in the allocated DM Setup Area. This instruction reads the data from the Serial Communications Unit as soon as the instruction is executed to achieve high-speed data reception.

Symbol

DRXDU



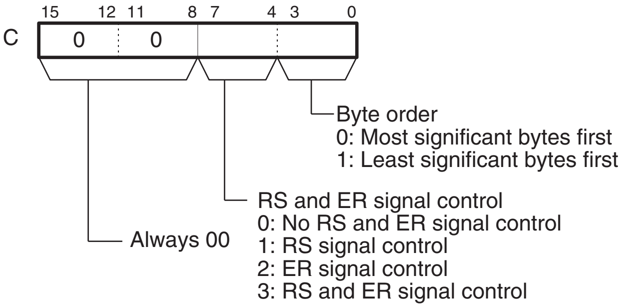
Applicable Program Areas

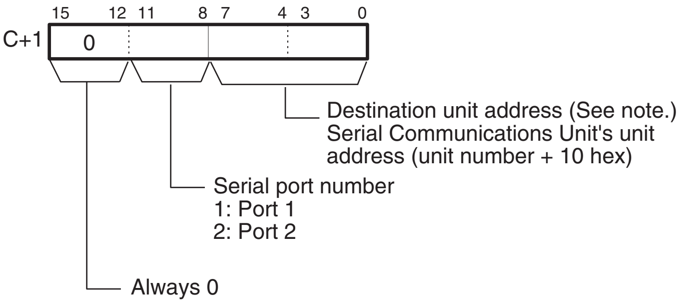
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	First destination word	UINT	Variable
C	First control word	UDINT	2
N	Number of bytes 0000 to 0100 hex (0 to 256)	UINT	1

C: Control word





● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

Name	Label	Operation
Error Flag	P_ER	• 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 execution of DTXDU(262)/DRXDU(261) in a cyclic task is interrupted by an interrupt task and DRXDU(261) is also executed in the interrupt task. • ON if the specified Unit is not a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit. (In this case, up to 1 ms will be required to execute the instruction.) • ON if the specified Serial Communications Unit is being initialized. • ON if the specified port on the Serial Communications Unit is not in No-protocol Mode. • OFF in all other cases.

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.
		0 to 7	Size of receive data	#00 hex (default value): 256 bytes #01 to FF hex: 1 to 256 bytes
m+25	m+35	8 to 15	Number of External Interrupt Task to Start for Interrupt Notification	00 to FF hex: 0 to 255
		5 to 7	Reserved.	---
		4	Interrupt Notification for CPU Unit for Data Reception	0: No interrupt notification 1: Interrupt notification
		1 to 3	Reserved.	---
		0	Reception Buffer Clear after DRXDU(261) Execution	0: Do not clear 1: Clear

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 RXD(235), RXDU(255), or DRXDU(261) 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) or DRXDU(261) was executed and has written the data from the buffer to a CPU Unit data area.
n+9	n+19	07	Reception Overflow Flag 0: The Board or Unit has not received more than the specified number of bytes. 1: The Board or Unit has received more than the specified number of bytes. 0 → 1:The Board or Unit received more data after data reception was completed. 1 → 0: RXDU(255) or DRXDU(261) was executed and has written the data from the buffer to a CPU Unit data area.
n+10	n+20	00 to 15	Reception Counter Indicates the number of bytes received in hexadecimal, between 0000 and 0100 hex (0 to 256 decimal).

Function

- DRXDU(261) reads data that has been received in no-protocol mode at the CJ1W-SCU22/32/42 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.
- This instruction reads the data from the Serial Communications Unit as soon as the instruction is executed to achieve high-speed data reception.
- The following receive-message frame formats can be set in the allocated DM Setup Area.
  1) Start code: None or 00 to FF hex.
  2) 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).
- Data will be stored in memory in the order specified in C.
- Cases where the Reception Completion Flag (*1) turns ON
  1) The Reception Completed Flag (note (a)) will turn ON when the number of bytes specified in the allocated DM Setup Area has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter (note (b)) will have the same value as the number of receive bytes specified in the allocated DM Setup Area.
  2) If an end code is specified in the allocated DM Setup Area, the Reception Completed Flag (note (a)) will turn ON when the end code is received or when 256 bytes of data have been received.
- Cases where the Reception Completion Flag (*1) turns ON
  1) If more data is received before RXDU(255) or DRXDU(261) is executed after the Reception Completed Flag (note (a)) turns ON, the Reception Overflow Flag (note (c)) will turn ON.
  2) If more bytes than specified in the allocated DM Setup Area are received than specified, the Reception Overflow Flag (note (c)) 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 (note (d)) and Transmission Error Flag (note (e)) will turn ON.
- When more data is input to the Serial Communications Board's serial port than is specified in N, that data will be cleared or will be maintained when the next DRXDU(261) instruction is executed depending on the setting of the Reception Buffer Clear after DRXDU(261) Execution parameter.
- When DRXDU(261) is executed, data is stored in memory starting at D, the Reception Completed Flag (note (a)) will turn OFF (even if the Reception Overflow Flag (note (c)) is ON), and the Reception Counter (note (b)) will be cleared to 0.

- Specification of monitor in bits 4 to 7 of C for the CS and DR signals takes effect as follows:
  1) If CS signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D.
  2) If DR signal monitoring is specified in C, the status of the DR signal will be stored in bit 15 of D.
  3) 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.
- If 1, 2, or 3 hex is specified for RS and DR signal control in C, DRXDU(261) will be executed regardless of the status of the Receive Completed Flag (note (a)).
- Receive data will not be stored if CS or DR signal monitoring is specified.
- Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.
- Specify the size of the receive data, not including the start code and end code, in N.
- Clearing the reception buffer immediately after DRXDU(261) execution can be enabled or disabled in the allocated DM Area words (m+25/m+35 bit 00).

Note Related Auxiliary Area and CIO Area Addresses

(a) Reception Completed Flags (n = CIO 1500 + 25 × unit number)

Port 1: Bit 6 of n+9

Port 2: Bit 6 of n+19

(b) Reception Counters (n = CIO 1500 + 25 × unit number)

Port 1: n+10

Port 2: n+20

(c) Reception Overflow Flags (n = CIO 1500 + 25 × unit number)

Port 1: Bit 7 of n+9

Port 2: Bit 7 of n+19

(d) Overrun Error Flags (n = CIO 1500 + 25 × unit number)

Port 1: Bit 4 of n+8

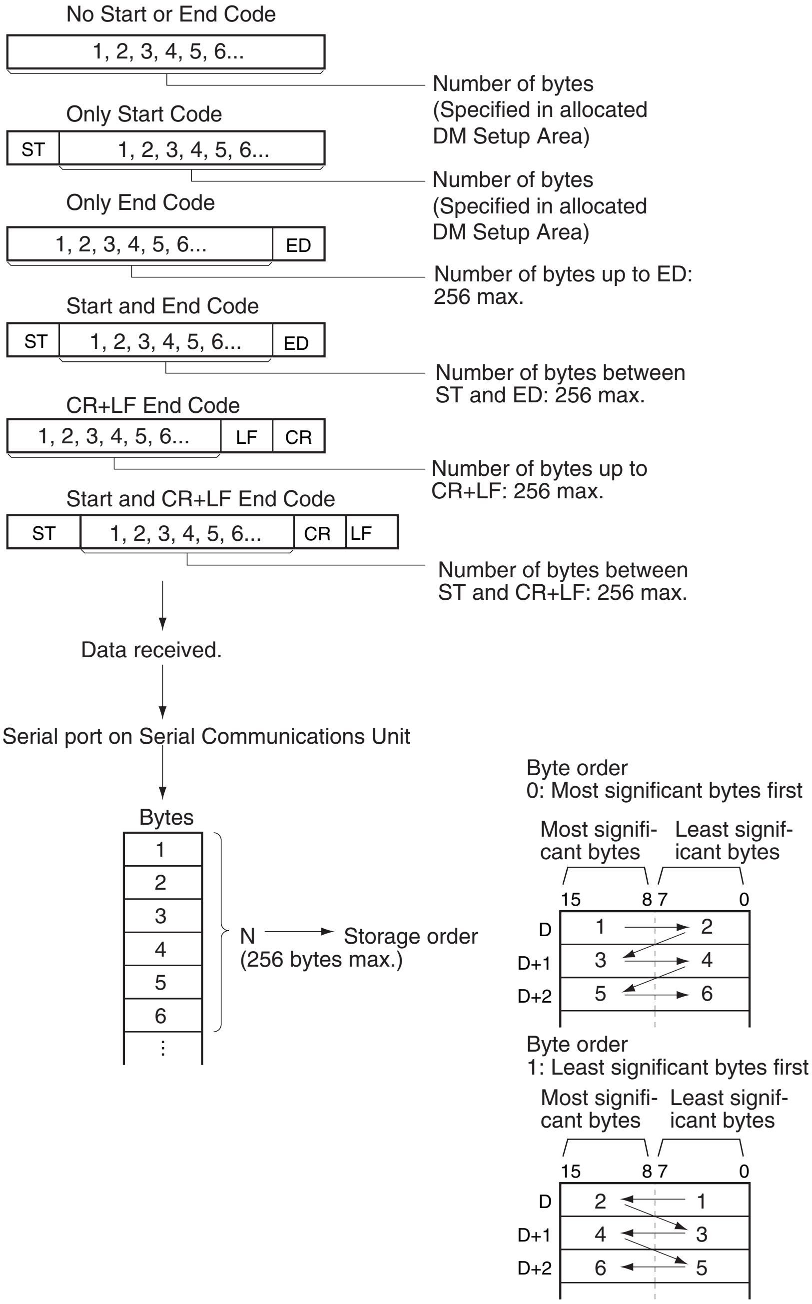
Port 2: Bit 4 of n+18

(e) Transmission Error Flags (n = CIO 1500 + 25 × unit number)

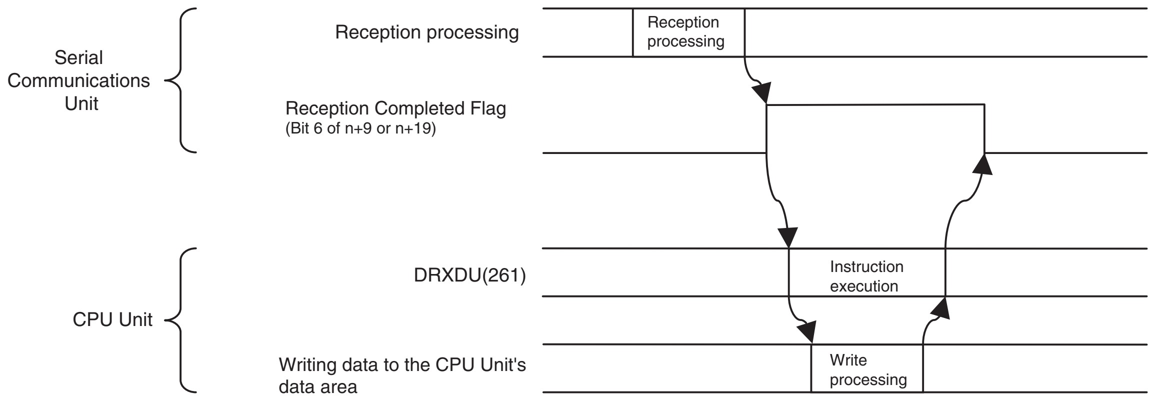
Port 1: Bit 15 of n+8

Port 2: Bit 15 of n+18

Start Code/End Code Settings and Send Data



Flag Operation



Hint

- Execute RXDU(255) to read the received data from the buffer when the Reception Completed Flag (in the allocated DM Setup Area) is ON.

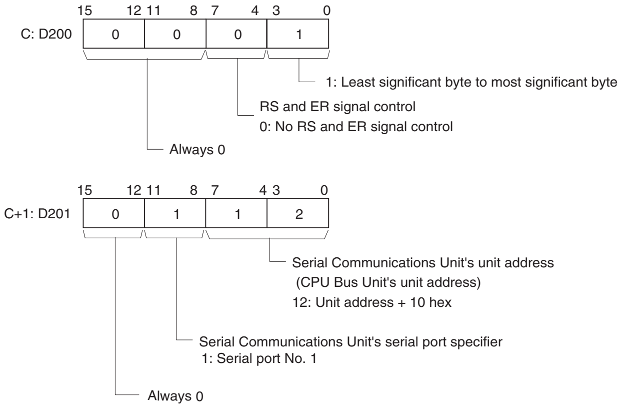
Precautions

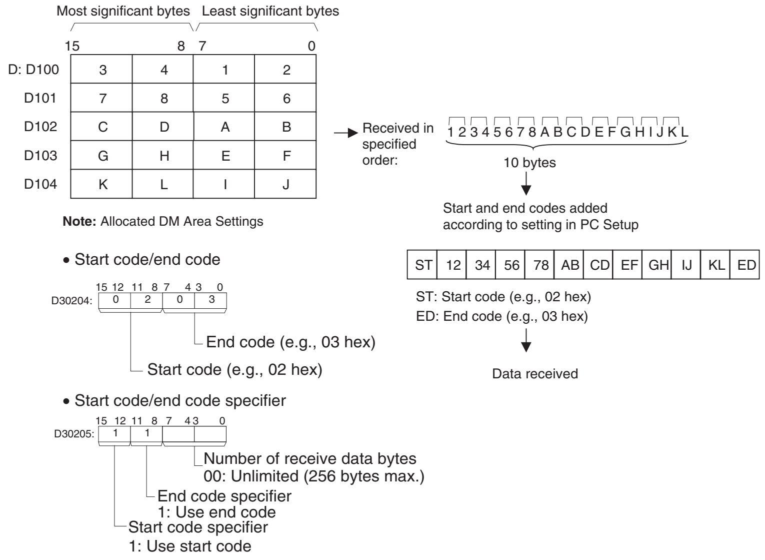
- DRXDU(261) can be executed only for serial ports on a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit. If it is executed for any other CPU Bus Unit, approximately 1 ms will be required to execute the instruction, increasing the cycle time.
- Further data cannot be received until the received data is read from the buffer with RXDU(255) or DRXDU(261). When the Reception Completed Flag goes ON, read that data promptly with RXDU(255) or DRXDU(261) before more data is input to the port.
- DRXDU(261) can be used only for serial ports set to No-protocol Mode on a CJ1W-SCU22/SCU32/SCU42 Serial Communications Unit mounted to a CJ2H CPU Unit with unit version 1.1 or later.
- 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.
- Program the Reception Completed Flag as a normally open condition of DRXDU(261) when executing it in a cyclic task. The Reception Completed Flag cannot be used when programming DRXDU(261) in an interrupt task. (If an external interrupt occurs, then data reception will have been completed, so the Reception Completed Flag is not required in an interrupt task.)
- Do not restart the Serial Communications Unit during interrupt notification from the Serial Communications Unit to the CPU Unit. Otherwise, system operation may not be stable.

Example Programming

Using DRXDU(261) in Cyclic Tasks

When CIO 0.00 is ON and CIO 1559.06 (the Reception Completed Flag for port 1) is ON in the following example, DRXDU(255) reads the data received from the general-purpose device connected to serial port 1 of the Serial Communications Unit with unit number 2. The 10 bytes of received data are written to the DM Area beginning at the rightmost byte of D100.





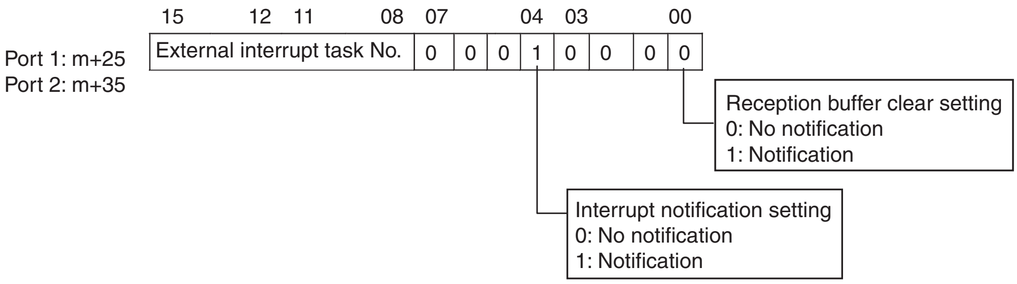
Using DRXDU(261) in Interrupt Tasks

This example shows how to start interrupt task 5 in the CPU Unit when 10 bytes of data is received from an external device (e.g., a barcode reader) connected to the Serial Communications Unit with unit number 2 and store the data in memory starting from the lower byte of D00100.

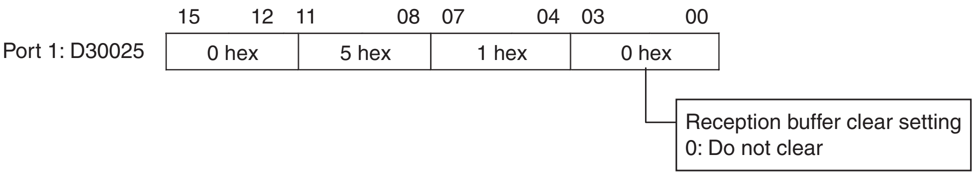
1. Settings in the DM Area Words Allocated to the Serial Communications Unit

   The following settings would be made in the allocated DM Area words m+25 or m+35.

m = D30000 + 100 × unit number

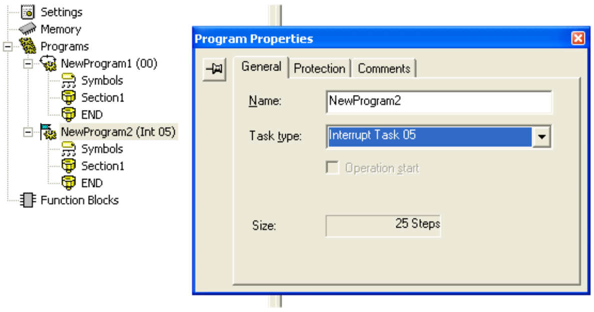


Example: The following settings would be made to use task 5 as the external interrupt task with the Serial Communications Unit with unit number 0.



The above settings in the allocated DM Area words can also be made by editing the CPU Bus Unit settings of the Serial Communications Unit in the I/O tables of the CX-Programmer.

2. The program to be assigned to the external interrupt task is right-clicked in the project tree in the CX-Programmer and ***Properties*** is selected. Interrupt task 05 is selected for the task type in this example.



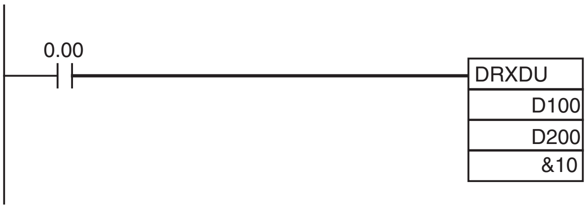
Name: NewProgram2

Task type: Interrupt Task 05

☐ Operation start

Size: 25 Steps

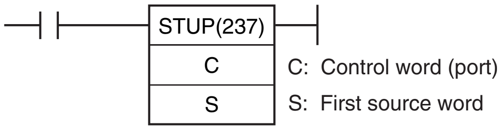
3. The following instructions are placed in the program assigned to the external interrupt task.



Note The Reception Completed Flag is not used in an interrupt task.

STUP

Instruction	Mnemonic	Variations	Function code	Function
CHANGE SERIAL PORT SETUP	STUP	@STUP	237	Changes the communications parameters of a serial port on the CPU Unit, Serial Communications Board (CS Series only), or Serial Communications Unit (CPU Bus Unit).

	STUP
Symbol	

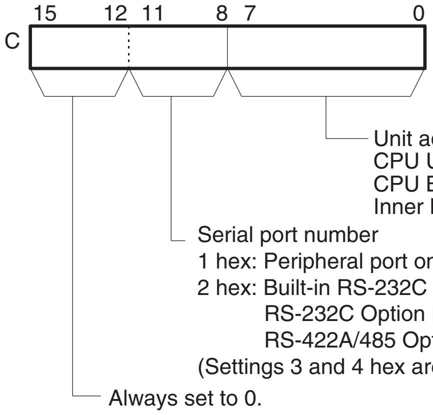
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
C	Control word (port)	UINT	1
S	First source word	UINT	10

C: Control word (port)



Unit address of port device
CPU Unit: 00 hex
CPU Bus Unit: Unit number + 10 hex
Inner Board: E1 hex (CS Series only)

Serial port number
1 hex: Peripheral port on CPU Unit or Port 1 on CPU Bus Unit or Inner Board
2 hex: Built-in RS-232C port on CPU Unit or Port 2 on CPU Bus Unit or Inner Board
RS-232C Option Board (CJ2M-CPU3□ only)
RS-422A/485 Option Board (CJ2M-CPU3□ only)
(Settings 3 and 4 hex are reserved.)

Always set to 0.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S												---						

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.

Related Auxiliary Area Words and Bits

Name	Address	Contents
Peripheral Port Parameters Changing Flag	A619.01	ON when the communications parameters are being changed for the peripheral port.
RS-232C Port Parameters Changing Flag	A619.02	ON when the communications parameters are being changed for the RS-232C port.
Port Parameters Changing Flags for ports 1 to 4 on Serial Communications Units 1 to 15.	A620 bit 01 to bit 04 to A635 bit 01 to bit 04	ON when the communications parameters are being changed for a port on a Serial Communications Unit.
Port Parameters Changing Flags for ports 1 to 4 on the Serial Communications Board (CS Series only)	A636.01 to A636.04	ON when the communications parameters are being changed for a port on the Serial Communications Board.

Note The Settings Changing Flags for port 3 and port 4 are reserved for future use.

Function

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	Peripheral Port	Communications parameters for the peripheral port in the PLC Setup (Cannot be specified for a CJ2 CPU Unit.)
		2 hex	RS-232C Port RS-232C Option Board RS-422A/485 Option Board	Communications parameters for the built-in RS-232C port, the RS-232C Option Board, and the RS-422A/485 Option Board in the PLC Setup.
Unit No. + 10 hex	Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1	10 words starting from D30000 + 100 × Unit No.
		2 hex	Port 2	10 words starting from D30000 + 100 × Unit No. + 10
E1 hex	Serial Communications Board (Inner Board) (CS Series only)	1 hex	Port 1	10 words starting from D32000
		2 hex	Port 2	10 words starting from D32010

Note The RS-232C Option Board and RS-422A/485 Option Board are only available on the CJ2M-CPU3□.

The following data is stored in the 10 words from S to S+9.

Peripheral port on CPU Unit	PLC Setup settings in Programming Console addresses +144 to +153
RS-232C port built into CPU Unit, RS-232C Option Board, RS-422A/485 Option Board	PLC Setup settings in Programming Console addresses +160 to +169[*1]
Serial Communications Unit port 1	m to m+9 (m = D30000 × unit number)
Serial Communications Unit port 2	m+10 to m+19 (m = D30000 ¥ unit number)
Serial Communications Board port 1	D32000 to D32009
Serial Communications Board port 2	D32010 to D32019

*1 Because the CJ2M does not have the Programming Console function, refer to the table below for the settings. The settings vary by communication mode. Settings with “Yes” exist.

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.

Note If the PLC is turned OFF and then ON again after STUP(237) has been used to change the communications parameters, the new parameters will be retained or will revert to the previous parameters, depending on the CPU Unit.

CPU Unit	Status of communications parameters
CJ2, CS1-H, CJ1-H, CJ1M, or CS1D	If the PLC is turned OFF and then ON again, the communications parameters revert to the settings that existed before they were changed with STUP(237).
CS1/CJ1	If the PLC is turned OFF and then ON again, the communications parameters set with STUP(237) are retained.

Word	Bit	Settings		Communication Mode						
				Host Link	NT Link	Peripheral Bus	No-protocol	Serial gateway	Serial PLC Link (master)	Serial PLC Link (slave)
+160	15	Any setting	0: Standard 1: Any setting	Yes*1		Yes*2	Yes*1	Yes*2		
	8 to 11	Communication Mode	0 hex: Host Link 2 hex: NT Link 3 hex: No-protocol 4 hex: Peripheral Bus 5 hex: Host Link 7 hex: Serial PLC Link Slave 8 hex: Serial PLC Link Master 9 hex: Serial gateway	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	3	Data length	0: 7 bits 1: 8 bits	Yes			Yes	Yes		
	2	Stop bits	0: 2 bits 1: 1 bit	Yes			Yes	Yes		
	0 to 1	Parity	00: Even 01: Odd 10: None	Yes			Yes	Yes		
+161	0 to 7	Port communication speed	00 hex: 9,600 01 hex: 300 02 hex: 600 03 hex: 1,200 04 hex: 2,400 05 hex: 4,800 06 hex: 9,600 07 hex: 19,200 08 hex: 38,400 09 hex: 57,600 0A hex: 115,200 Units: Bits/sec	Yes	Yes*3	Yes*4	Yes	Yes	Yes*3	Yes*3
+162	0 to 15	Transmission delay	0000 to 270F hex: 0 to 99,990 ms (Units of 10 ms)				Yes			
+163	0 to 7	Host Link Unit No.	00 hex to 1F hex: 0 to 31	Yes						
+164	8 to 15	Start Code	00 to FF hex				Yes			
	0 to 7	End Code	00 to FF hex				Yes			
+165	12	Start Code Specifier	0: None 1: Use Start Code				Yes			
	8 to 9	End Code Specifier	0 hex: None (number of Bytes of Data) 1 hex: Use End Code 2 hex: Set end code to CR+LF				Yes			
	0 to 7	Number of Bytes of Data	00 hex: 256 01 to FF hex: 1 to 255				Yes			
+166	15	Link method	0: All station link method 1: Master link method						Yes	
	4 to 7	Number of link words	1 to A hex						Yes	
	0 to 3	Maximum unit number	0 to 7 hex		Yes				Yes	
+167	8 to 15	Response monitoring time	00 hex: 5 s 01 to FF hex: 100 to 25,500 ms (Units of 100 ms)					Yes		
	0 to 3	Slave unit number	0 to 7 hex							Yes

*1 If "0: Standard" is selected when host link and no-protocol are specified, the settings will be as follows:
- Start bit: 1 bit
- Data length: 7 bits
- Parity: Even
- Stop bit: 2 bits
- Communication speed: 9,600 bits/sec

*2 If "0: Standard" is selected when the peripheral bus and serial gateway are specified, the settings will be as follows:
- Communication speed: 9,600 bits/sec

*3 If NT Link, serial PLC Link (master), and serial PLC Link (slave) are specified, the 00 hex and 0A hex settings are enabled.

*4 If the peripheral bus is specified, the 00 hex and 06 hex to 0A hex settings are enabled.

Hint

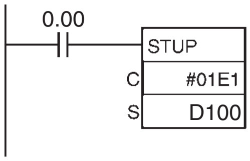
- Use STUP(237) to change communications parameter for a port during operation based on specified conditions. For example, STUP(237) can be used to change to Host Link communications for monitoring and programming from a host computer when specified conditions are meet while execution a communications sequence for a modem connection.
- 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 the following manuals for details.
  - SYSMAC CS/CJ/NSJ Series Programmable Controllers Programming Manual (W394) “6-3 Serial Communications”
  - SYSMAC CJ Series Programmable Controllers Operation Manual (W393) “7-1 PLC Setup”
  - SYSMAC CS/CJ Series Serial Communications Boards/Units Operation Manual (W336)
  - CJ-series CJ2 CPU Unit Software User’s Manual (W473)

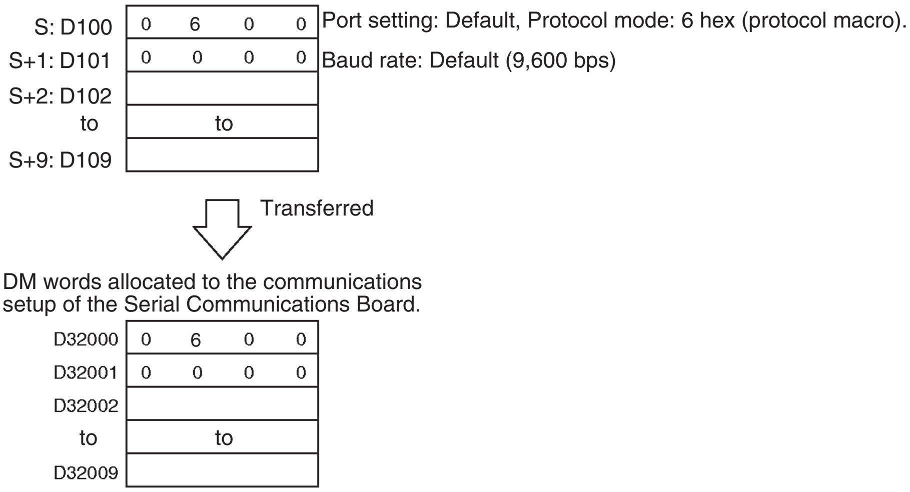
    “4-2 PLC Setup”

    “11-2 Serial Communications”

Example Programming

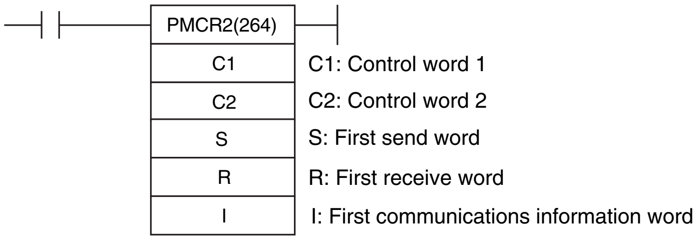
When CIO 0.00 turns ON in the following example, the communications parameters for serial port 1 of the Serial Communications Board (Inner Board) are changed to the settings contained in the 10 words from D100 to D109. In this example, the setting are changed from the protocol mode to the protocol macro mode.





PMCR2

Instruction	Mnemonic	Variations	Function code	Function
PROTOCOL MACRO 2	PMCR2	@PMCR2	264	Starts a communications sequence (protocol data) that is registered in a Serial Communications Unit.

	PMCR2
Symbol	

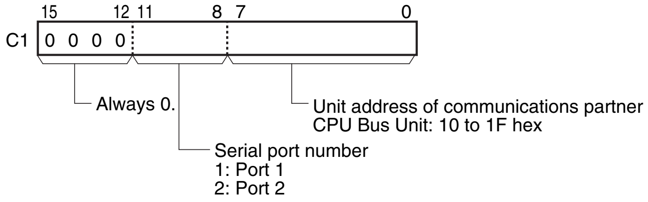
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

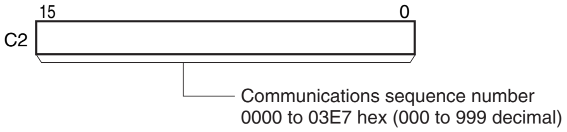
Operands

Operand	Description	Data type	Size
C1	Control word 1	WORD	1
C2	Control word 2	WORD	1
S	Control word 1	WORD	Variable
R	First receive word	WORD	Variable
I	First communications information word	WORD	2

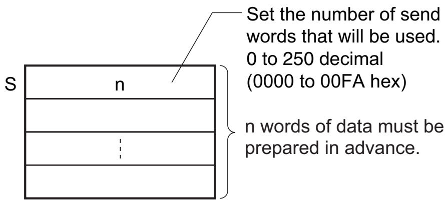
C: Control word 1



C: Control word 2



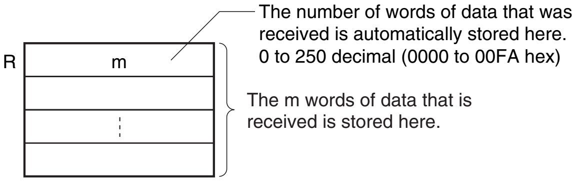
● Send Area Specification S: First send word

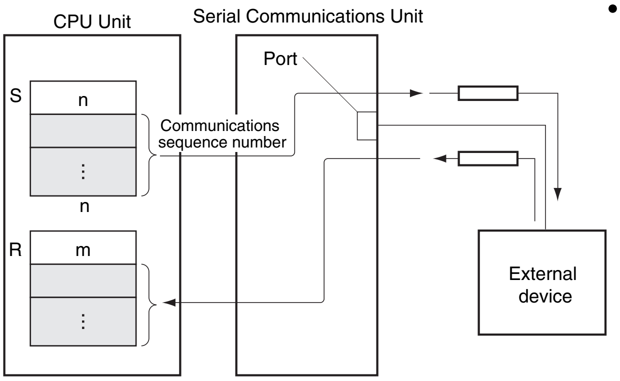


If there is no operand specified in the execution sequence, such as a direct or linked word, specify the constant 0000 hex for S. If a word address or register is specified, the data in the word or register must always be 0000 hex. An error will occur and the Error Flag will turn ON if any other constant or a word address is given and PMCR2(264) will not be executed.

● Receive Area Specification
R: First receive word

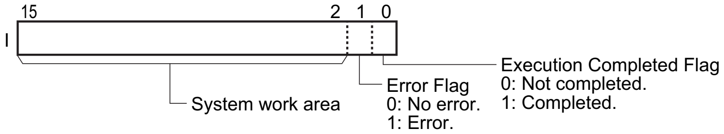
- Executing a Protocol Macro

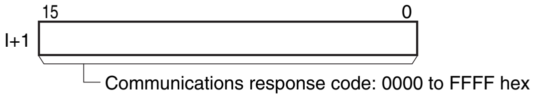




- Setting before Executing PMCR2
  Set the data specified by m (beginning with R) as the initial data for the receive buffer (backup data for receive failure).
  Data m can be set to 2 to 250 decimal (0002 to 00FA hex). If 0000 or 0001 hex is specified for m, the initial value of the receive buffer will be cleared to 0.
- If there is no operand specified in the execution sequence, such as a direct or linked word, specify the constant 0000 hex for R. If a word address or register is specified, the data in the word or register must always be 0000 hex.

I: First communications information word





● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
C1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C2																		
S																		
R																		
I											---	---						

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if C1 or C2 is not within the specified ranges. • ON if the number of words of S or R exceeds 249 (when word addresses are specified). • ON if there is no communications port (i.e., internal logical port) available (i.e., if A211 = 0). • OFF in all other cases.

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.

Note Refer to the *CS/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	

Function

PMCR2 calls and executes a communications sequence (protocol data) registered in a Serial Communications Board or Serial Communications Unit. PMCR2 will execute the communications sequence specified in C2 using the port specified in 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 starting with R+1 and the number of words received is automatically written to R if the transmission is successful.

When the transmission starts, the Execution Completed Flag in I is turned OFF. When a response is returned, the transmission result is shown in the Error Flag in I. The communications response is written to I+1.

Hint

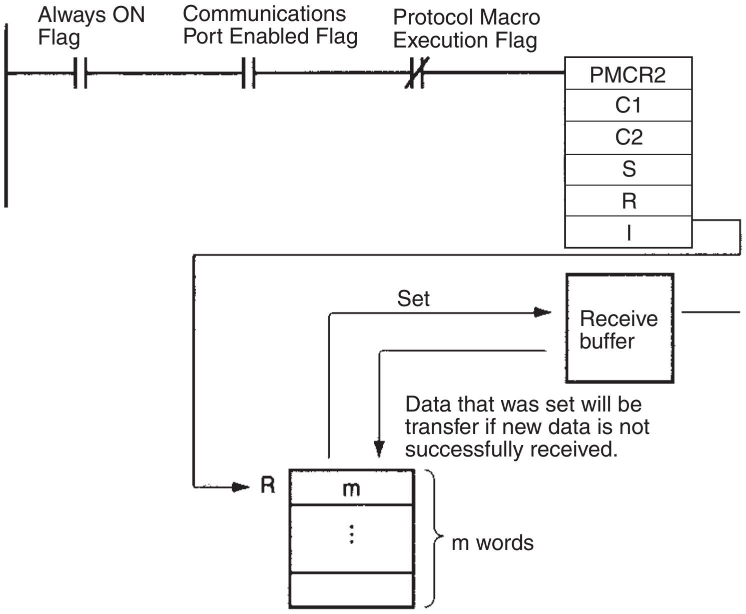
● Holding the Receive Area

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.

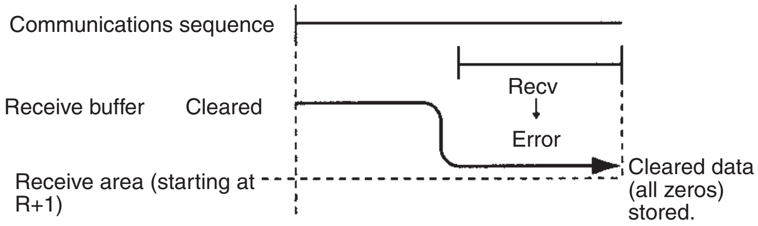
A function is provided to maintain the data in the receive area even when a reception error occurs. If this function is used, data will be transferred from the first m words of the receive area to the receive buffer after the buffer is cleared to all zeros but before the communications sequence is executed. 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.

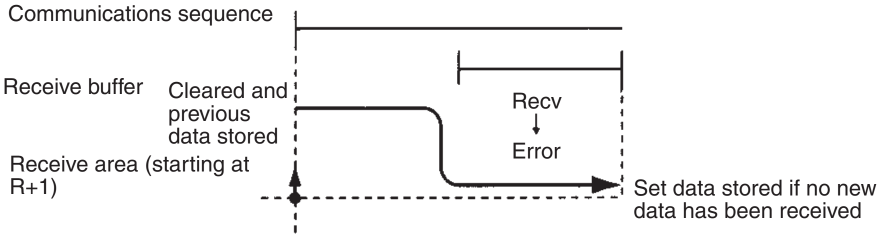
Example: The following programming example shows the instructions used to constantly or periodically execute PMCR2(264) to read data through a single receive operation.



Receive Area Not Held



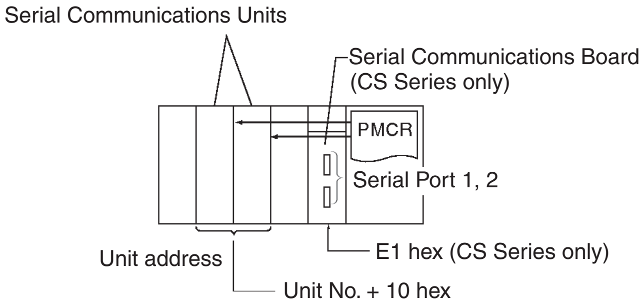
Receive Area Held



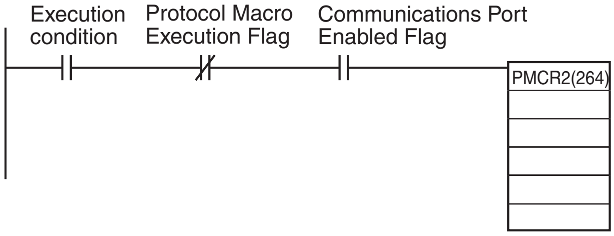
Precautions

- 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( ).
- PMCR2(264) can be executed for a serial communications port on a Serial Communications Unit. Up to 16 Serial Communications Units can be mounted to the CPU Rack and Expansion I/O Racks. The Unit address of the communications partner must be set in bits 0 to 7 of C1 to specify which Unit/Board 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 PMCR2(264) 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 PMCR2(264) 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.



- An error flag will not turn ON when the data of C2 is outside the range. A completion code will be stored in “Network Communications Completion Code” (A203 to A210) of the Auxiliary.

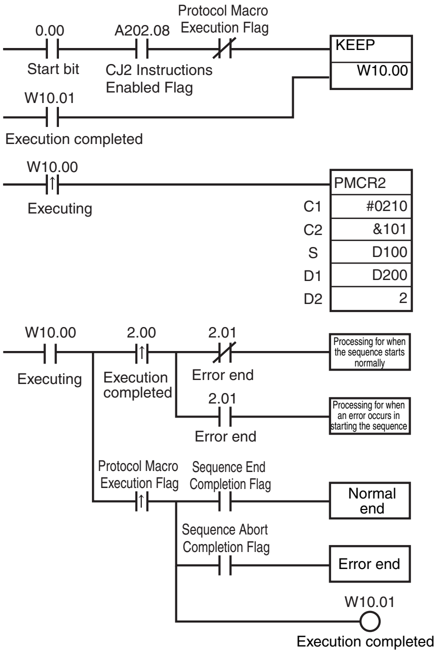
Example Programming

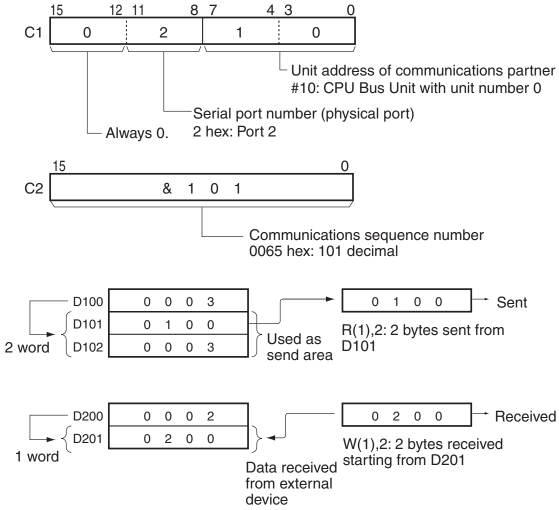
If CIO 0.00 (start bit) and A202.08 (CJ2 Instructions Enabled Flag) are ON, W10.00 (executing) will turn ON. When W10.00 (executing) turns ON, communications sequence 101 (101 decimal or 0065 hex) is executed. If a symbol is specified as the operand for the send message, the two words beginning from D101 will be used as the send area (D100 contains 0003 hex).

If a symbol is specified as the operand for the receive message, receive data is placed in memory starting with D201 and the number of words received (counting D200) is automatically written to D200 if the transmission is successful.

If communications end normally, CIO 2.00 (execution completed) turns ON, and the processing for when the sequence starts normally is executed. If communications end in an error, CIO 2.01 (error end) turns ON, and the processing for when an error occurs in starting the sequence is executed.

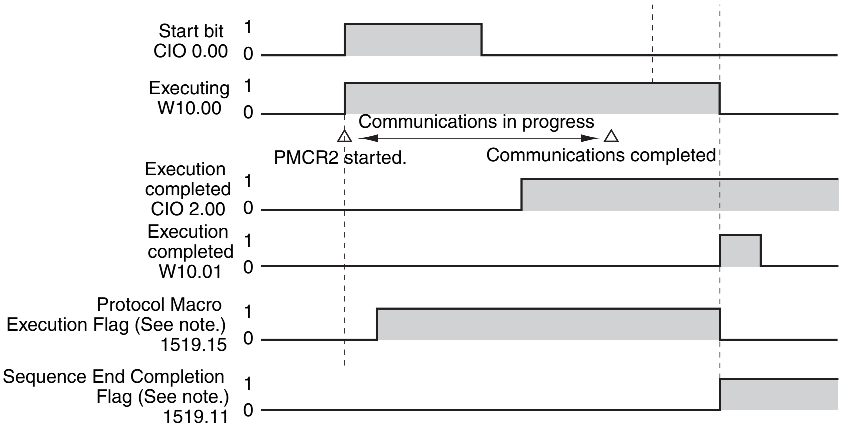
Note PMCR2(264) is used to start a communications sequence. Use the Sequence End Completion Flag in the CIO Area words that are allocated to the Serial Communications Unit to determine when execution of the communications sequence has ended normally.





Note As shown above, the symbol read option, R( ), in the send message or the symbol write option, W( ), in the receive message, actually sends/receives data

● Timing Chart



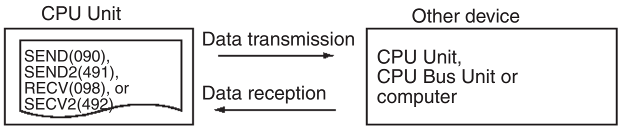
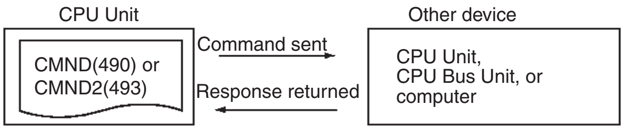
Note Refer to the *SYSMAC CS/CJ-series Serial Communications Boards and Serial Communications Units Operation Manual* (Cat. No. W336) for details on the CIO Area words that are allocated to the Serial Communications Unit.

Network Instructions

Network Instructions

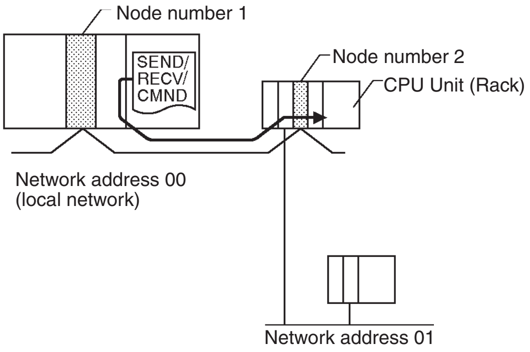
Overview of Network Communications Instructions

The network instructions can be divided into two types, SEND(090)/SEND2(491)/RECV(098)/RECV2(492) and CMND(490)/CMND2(493). These instructions are transmitted between Units (CPU Units, CPU Bus Units, and computers) in a network to transfer data and control operation, such as changing the operating mode.

Message content	Instruction	Operation
Commands to transmit/receive data (FINS command)	SEND(090)/SEND2(491) RECV(098)/RECV2(492)	
Arbitrary commands (FINS command)	CMND(490)/CMND2(493)	

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 Series Communications Commands Reference Manual* (Cat. No. W342) for details on FINS commands.) With FINS commands it is possible to communicate (by the command/response format) with any Unit in any network or on the CPU Rack itself 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.

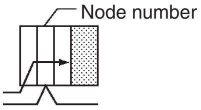
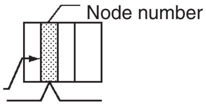
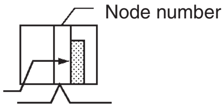
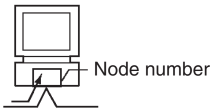


Network address: Address of the network (local network = 00 hex)

Node number: Logical address in the network

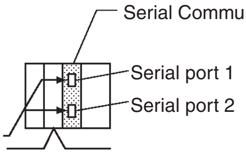
Unit address: Unit address of the destination Unit

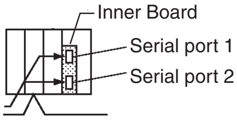
- CPU Unit: 00 hex
- CPU Bus Unit: Unit number + 10 hex
- Special I/O Unit: Unit number + 20 hex
- Inner Board: E1 hex
- Computer: 01 hex

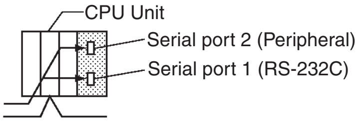
	CPU Unit	CPU Bus Unit	Inner Board	Computer
Unit address	**00 hex**	**Unit number +10 hex**	**E1 hex**	**01 hex**
Destination device				

Unit Addresses

It is also possible to directly specify a serial port (port 1 to 4) within the destination device.







Serial Port Unit Addresses:

- Serial Communications Unit ports

  Port 1: 80 + 4 × unit number (hex)

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	AB	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 + 4 × unit number (hex)

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

- Serial Communications Board ports

  Port 1: E4 hex (228 decimal)

  Port 2: E5 hex (229 decimal)

- CPU Unit ports

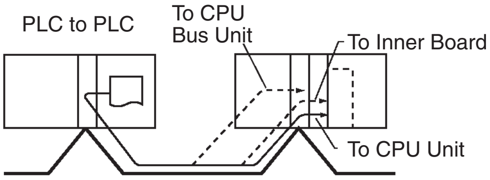
  Peripheral port: FD hex (253 decimal)

  RS-232C port: FC hex (252 decimal)

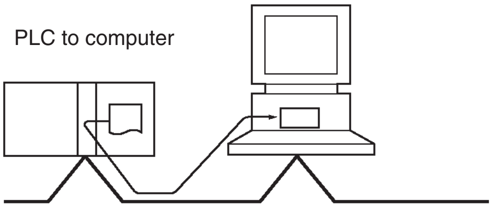
Network Communications Patterns

Communications with Another Device on 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.

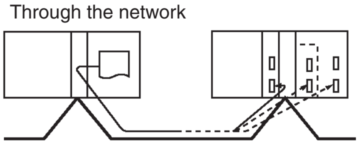


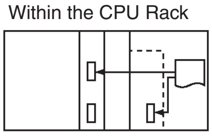
This example shows communications from a PLC to a personal computer.



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.

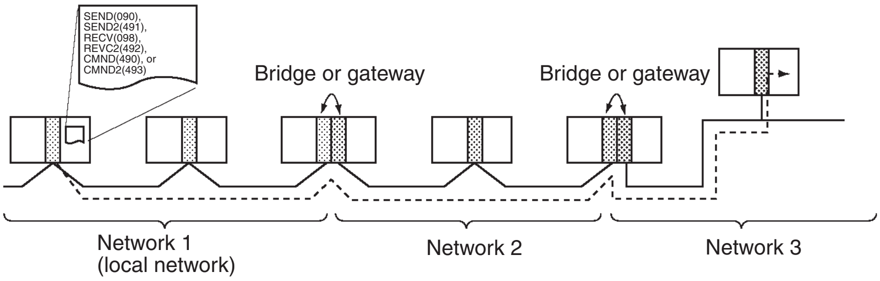




For details, refer to the operation manual for each network (Controller Link or Ethernet).

● Communicating with Devices on Other Networks

Communications can span up to 8 network levels, including the local network. (The local network is the network where the communications originate.)

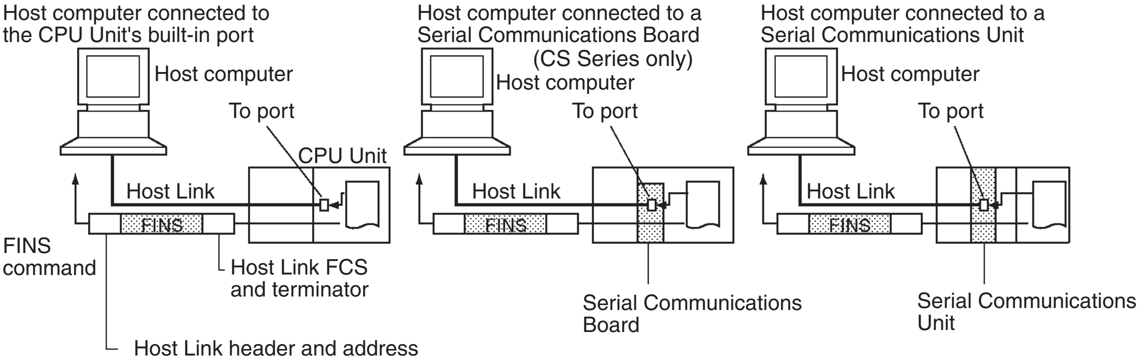


In order to communicate through the network, it is necessary to register a routing table in each PLC's CPU Unit which indicates the route by which data will be transferred to the desired node. Each routing table is made up of a local network table and a relay network table.

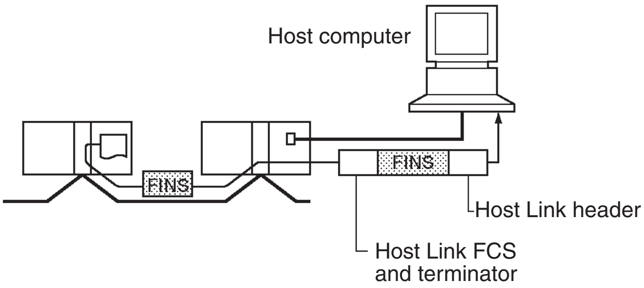
- Local network table: This table shows the unit numbers and network addresses of the nodes connected to the local PLC.
- Relay network table: This table shows the node numbers and network addresses of the first relay nodes to destination networks that are not connected to the local PLC.

Unsolicited Communications with a Host Computer Connected via Host Link

By executing a SEND(090), SEND2(491), RECV(098), RECV2(492), CMND(490), or CMND2(493) instruction for 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.

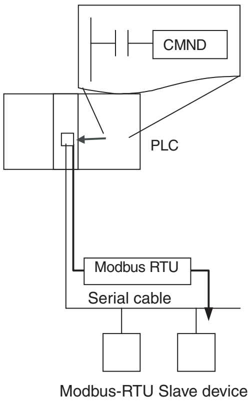


Serial Gateway Communications with a Component or Host Link Slave

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.

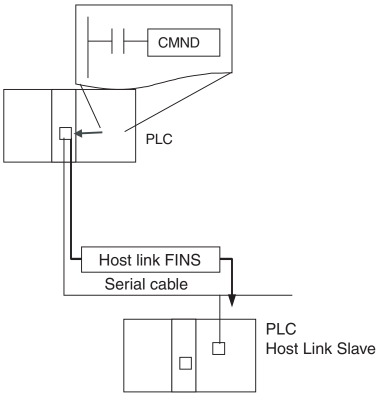
- Sending to a Component

  When a CMND(490) or CMND2(493) 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.



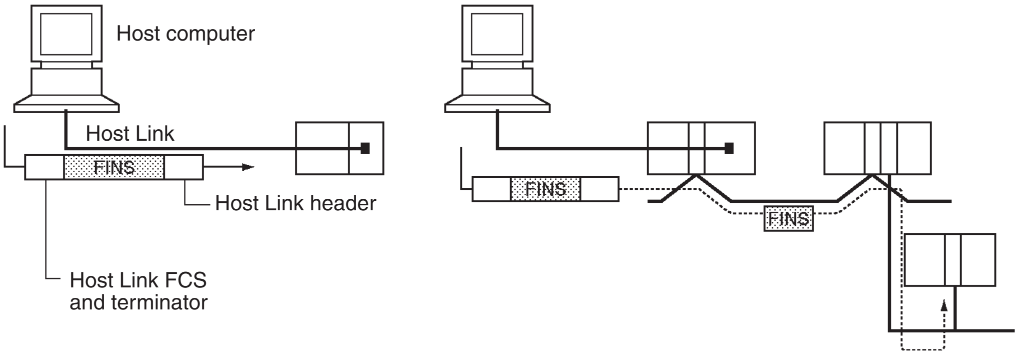
- Sending to a PLC Operating as a Host Link Slave

  When a CMND(490), CMND2(493), SEND(090), SEND2(491), RECV(098), or RECV2(492) 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 Connected through 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.



Network Communications Instructions

With CS/CJ-series PLCs, up to eight Network Communications Instructions can be executed simultaneously. With CJ2-series PLCs, up to 64 Network Communications Instructions can be executed simultaneously for some instructions. These instructions are called CJ2 Network Communications Instructions.

	Mnemonic	Instruction	Features
CJ2 Network Communications Instructions	SEND2	NETWORK SEND 2	Up to 64 of these instructions can be executed simultaneously. Words specified in the instruction are used to show the completion of communications and communications errors.
	RECV2	NETWORK RECEIVE 2	
	CMND2	DELIVER COMMAND 2	
	PMCR2	PROTOCOL MACRO 2	
Network Communications Instructions for CJ1/CS1/CJ2	SEND	NETWORK SEND	Up to 8 of these instructions can be executed simultaneously. Bits in the Auxiliary Area are used to show the completion of communications and communications errors. Either the communications port must be specified, or automatic allocation of a communications port must be specified. (Here, "communications port" indicates an internal logical port, not a physical port.)
	RECV	NETWORK RECEIVE	
	CMND	DELIVER COMMAND	
	EXPLT	EXPLICIT MESSAGE SEND	
	EGATR	EXPLICIT GET ATTRIBUTE	
	ESATR	EXPLICIT SET ATTRIBUTE	
	ECHRD	EXPLICIT WORD READ	
	ECHWR	EXPLICIT WORD WRITE	
	PMCR	PROTOCOL MACRO	
	TXDU	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	
	RXDU	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	

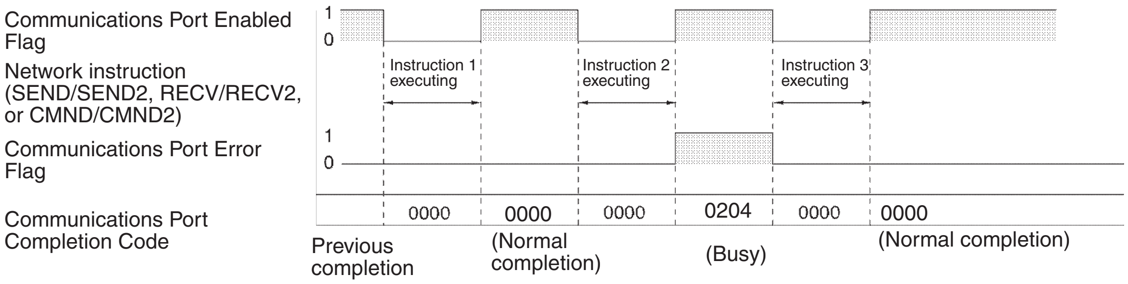
Communications Flags

The operation of the communications flags is outlined below.

- The Communications Port Enabled Flag is turned OFF when communications are in progress and turned ON 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 turned OFF the next time that data is transmitted or received, even if there was an error in the previous operation.

Note Here, “communications port” indicates an internal logical port, not a physical port.

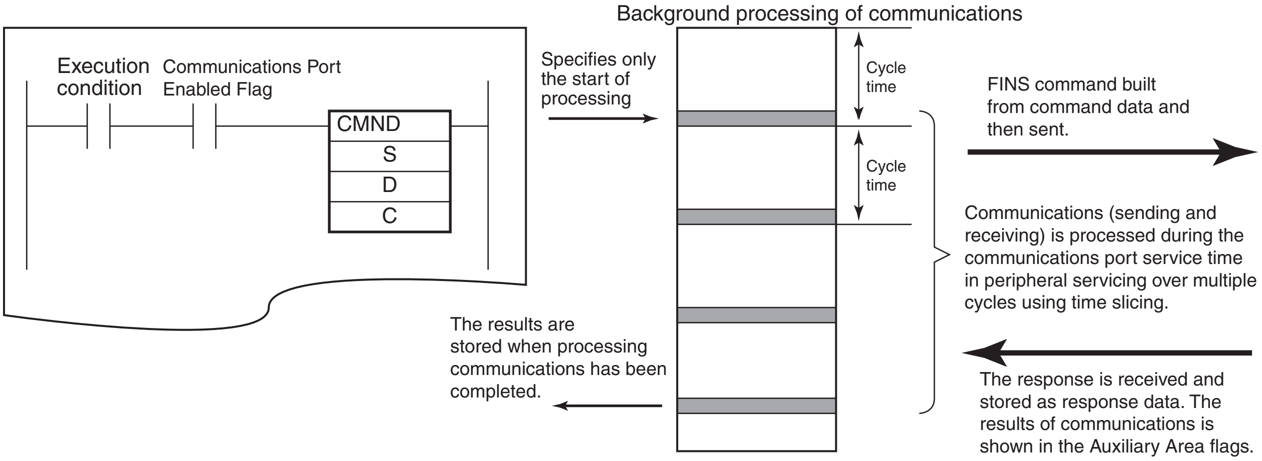
Example



Address	Bits	Name	Description
A202	00 to 07	Communications Port Enabled Flags	Turns ON when a communications instruction can be executed. Bits 00 to 07 correspond to communications ports 0 to 7. You can tell when communications have been completed because the Communications Port Enabled Flag will turn ON again after the completing execution of the communications instruction. The Communications Port Enabled Flag will be OFF while the communications instruction is being executed.
	15	Communications Port Allocation Enabled Flag	Turns ON when there is a communications port available for automatic allocation. **Note** Use this flag to see if all eight communications ports have already been allocated before executing communications instructions.
A203 to A210	---	Communications Port Completion Codes	These words contain the completion codes for the corresponding port numbers when network instructions have been executed.
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. **Note** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle. **Note** Use the Used Communications Port Number stored in A218 to determine which flag to access.
	08 to 15	Do not use.	---
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. If the flag turns ON, refer to the Communications Port Completion Code (A203 to A210) to identify the cause of the error. **Note** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle. **Note** Use the Used Communications Port Number stored in A218 to determine which flag to access.
	08 to 15	Do not use.	---
A216 and A217	---	Communications Port Completion Code Storage Address	The completion code for a communications instruction is automatically stored at the address with the PLC I/O memory address given in these words. **Note** 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.
A219	00 to 07	Communications Port Error Flags	Turns ON when a communications error occurs during execution of network communications. If the Communications Port Error Flag turns ON, refer to the Communications Port Completion Code in (A203 to A210) to identify the cause of the error. Bits 00 to 07 correspond to communications ports 0 to 7. The Communications Port Error Flag will be OFF while the communications instruction is being executed.

Execution Timing of Network Communications Instructions

When the execution condition for a Network Communications Instruction is ON, processing communications is started, but actual communications are processed in the background using peripheral servicing.

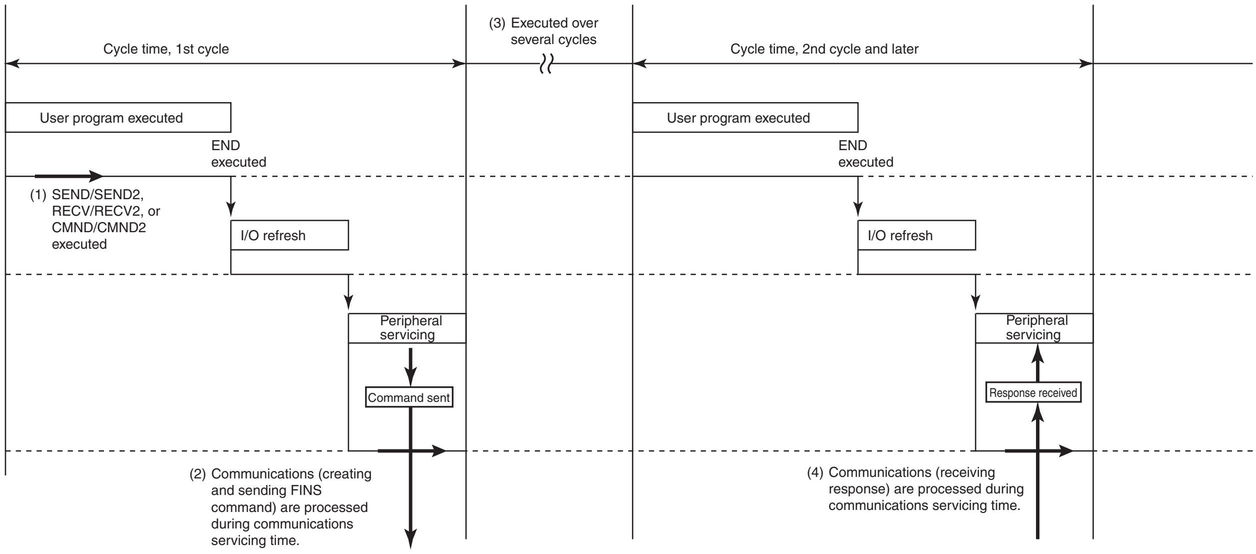


The following operation occurs.

(1) If the relevant Communications Port Enabled Flag (A202.00 to A202.07) is ON when the execution condition turns ON, it turns OFF, the Communications Port Error Flag (A219.00 to A219.07) turns OFF, the Communications Port Completion Codes (A203 to A210) is cleared to 0000 hex, the contents of the control words starting from C are read, and communications processing (sending the FINS command and receiving the response) is started.

(2) The communications command is processed during peripheral servicing. This processing lasts for the required number of cycles.

(3) When a response is received, the response words specified by the operand are updated in communications port servicing. The Communications Port Enabled Flag (A202.00 to A202.07) is turned ON, the Communications Port Error Flag (A219.00 to A219.07) is turned ON or OFF according to the results of the instruction, and the Communications Port Completion Code (A203 to A210) is stored.

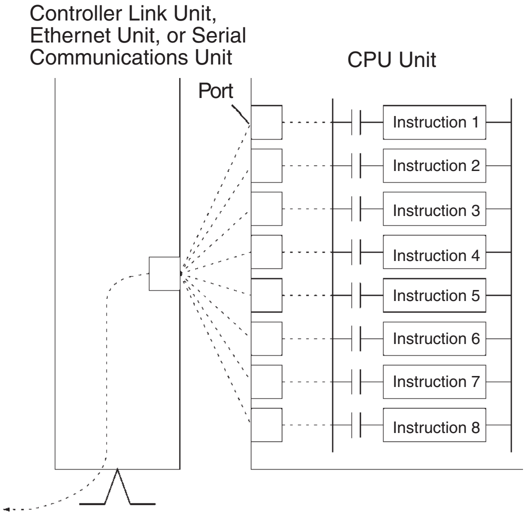


Network Communications Instructions for CJ1/CS1/CJ2

Communications Ports

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 ports are shared by the network instructions (SEND(090), SEND2(491), RECV(098), RECV2(492), CMND(490), CMND2(493), etc.), the serial communications instructions (TXDU(256) and RXDU(255)), and the PROTOCOL MACRO instruction (PMCR(260) and PMCR2(264)). Be sure not to specify the same port for two instructions at the same time.

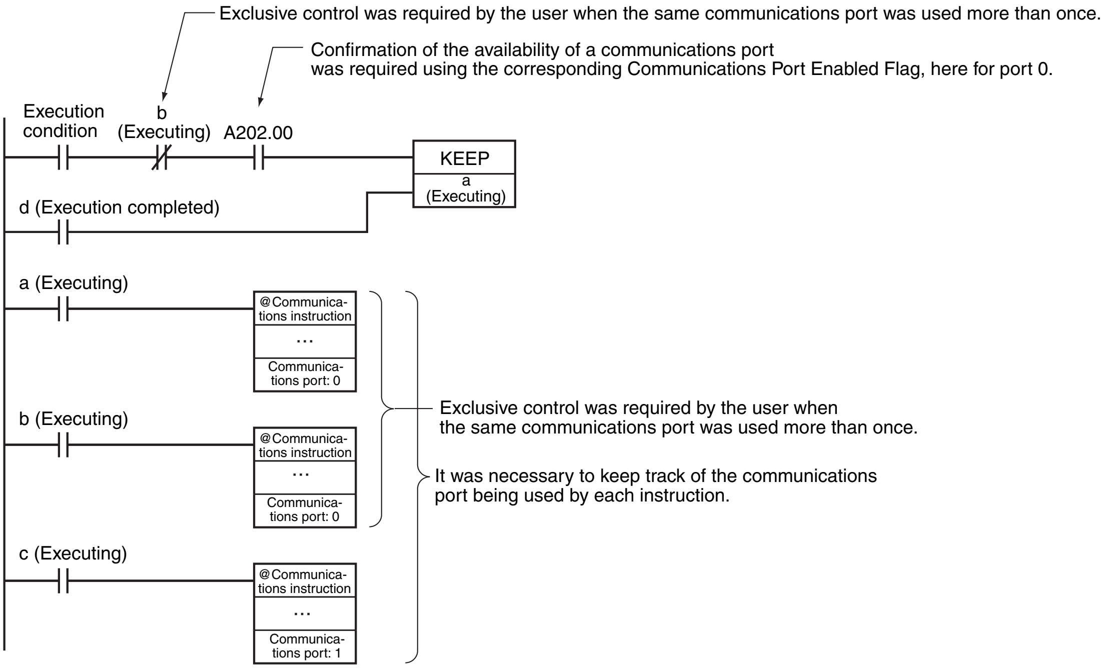


Network Communications Instructions for the CJ1/CS1/CJ2 can be executed either by specifying a communications port or by allowing the CPU Unit to automatically assign a port.

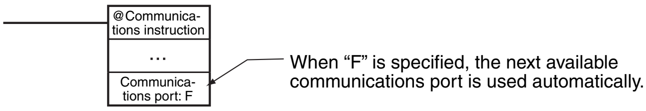
Each communications port can be used by only one instruction at a time. To specify a communications port, the following steps were necessary.

- When programming, it was necessary to keep track of the ports that were being used to designate only available ports in operands.
- In the ladder program, it was necessary to add processing to confirm the availability of communications ports before using them.

Example When Not Using Automatic Port Allocation



For CJ2 CPU Units and CS1-H, CJ1-H, CJ1M, and CS1D (Single CPU System) CPU Units of lot number 020601 or later (manufactured 1 June 2002 or later), 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.

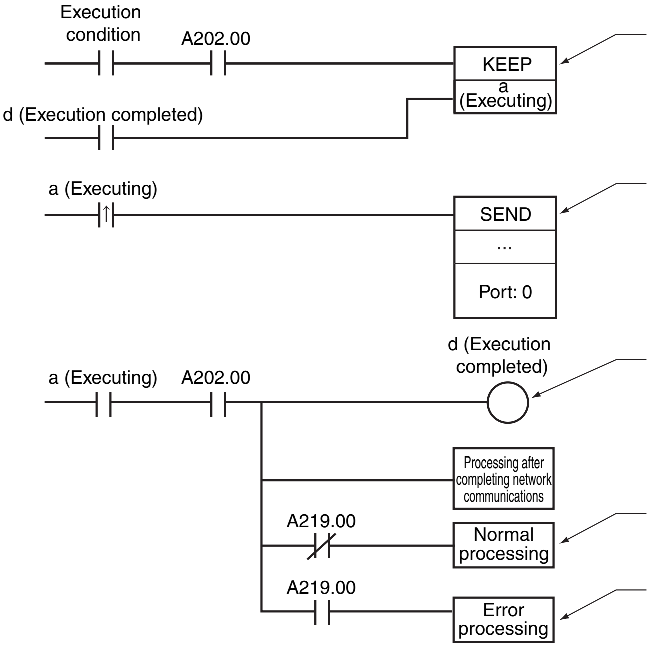


This saves the programmer from having to keep track of communications ports while programming. The differences between assigning specific port numbers and automatically allocating port numbers are given in the following table.

Item	User specification of communications ports	Automatic allocation of communications ports
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).
Communications Port Completion Codes	Completion code for communications port specified by user is accessed.	Completion codes are accessed by using the PLC memory address stored in A216 and A217 (Communications Port Completion Code Storage Address) and index register indirect addressing.

Ladder Programming Examples

● Specifying Communications Ports



a (Executing) is turned ON and held ON when the execution condition and Communications Port Enabled Flag (A202.00) turn ON.
a (Executing) is turned OFF when d (Execution completed) turns ON

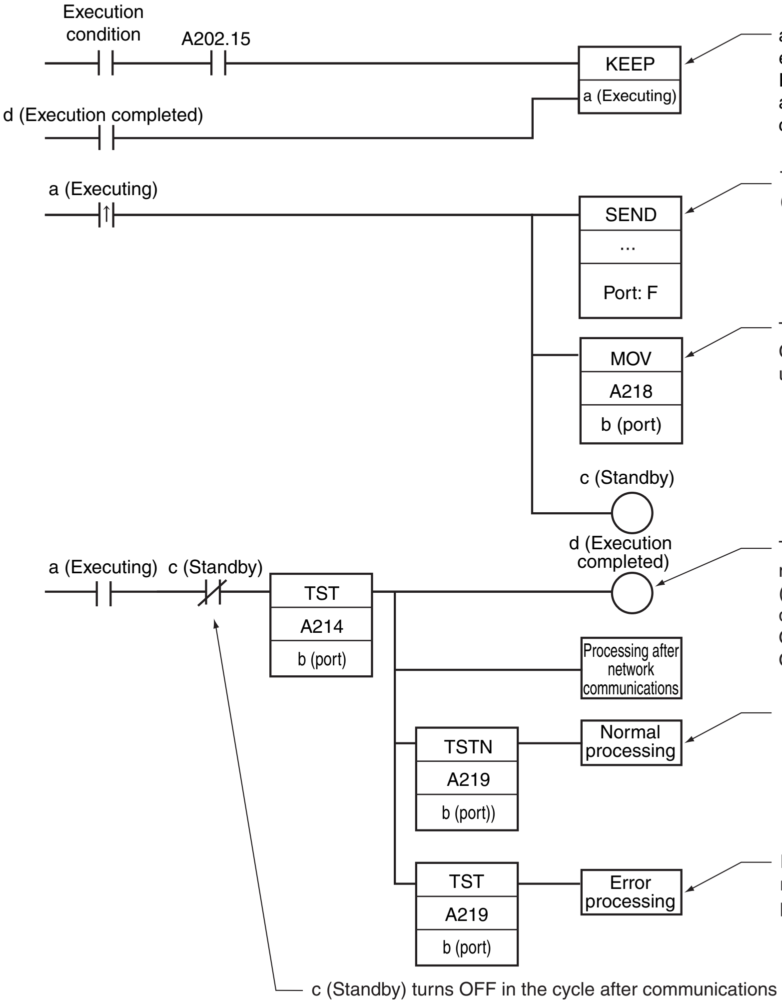
The network communications instruction is executed when a (Executing) turns ON. The port number 0 is specified.

The Communications Port Enabled Flag (A202.00) is OFF during network communications and turns ON when they are completed (for either normal or error completion).
d (Execution completed) is turned ON when a (Executing) turns ON and the Communications Port Enabled Flag (A202.00) turns ON.

If the Communications Port Error Flag (A219.00) is OFF when network communications have been completed, normal processing is performed.

If the Communications Port Error Flag (A219.00) is ON when network communications have been completed, error processing is performed.

● Automatically Allocating Communications Port



a (Executing) is turned ON and held ON when the execution condition and Communications Port Allocation Enabled Flag (A202.15) turn ON.
a (Executing) is turned OFF when d (Execution completed) turns ON.

The network communications instruction is executed when a (Executing) turns ON. The port number F is specified.

The automatically allocated port number stored in A218 (Used Communications Port Numbers 0 to 7) is moved to a user-specified work word b (port).

The Communications Port Enabled Flag (A202.00) is OFF during network communications and turns ON when they are completed (for either normal or error completion).
d (Execution completed) is turned ON when a (Executing) turns ON and the Communications Port Enabled Flag (A202.00) turns ON.

If the Communications Port Error Flag (A219.00) is OFF when network communications have been completed, normal processing is performed.

If the Communications Port Error Flag (A219.00) is ON when network communications have been completed, error processing is performed.

c (Standby) turns OFF in the cycle after communications instruction execution is completed and communications completion are detected from that cycle.

User Specification of Communications Ports

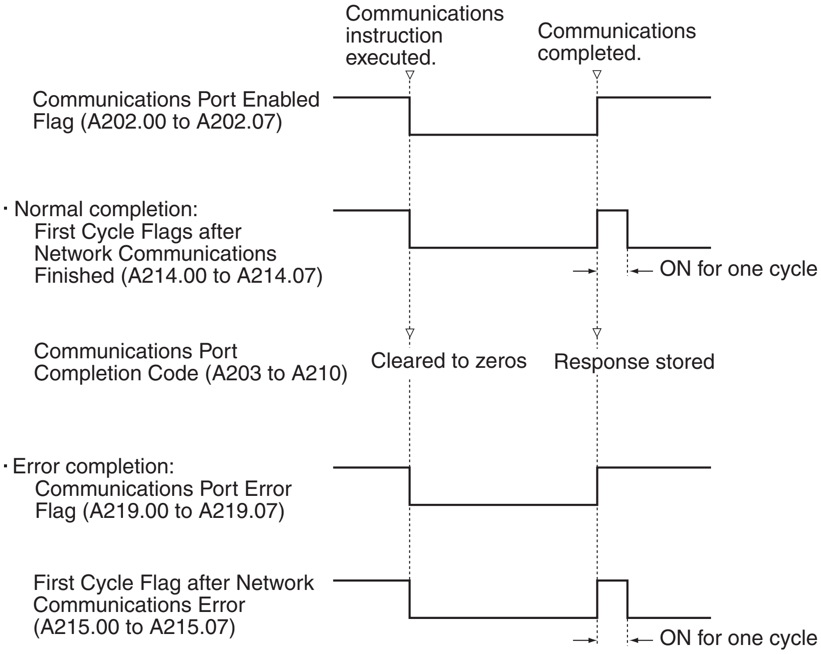
The communications port can be specified in the control data of the Network Communications Instruction.

The same communications port cannot be used in any other instruction until processing this instruction has been completed. Exclusive control of the communications port is necessary.

● Auxiliary Area Words and Bits Used for User Specification of Communications Ports

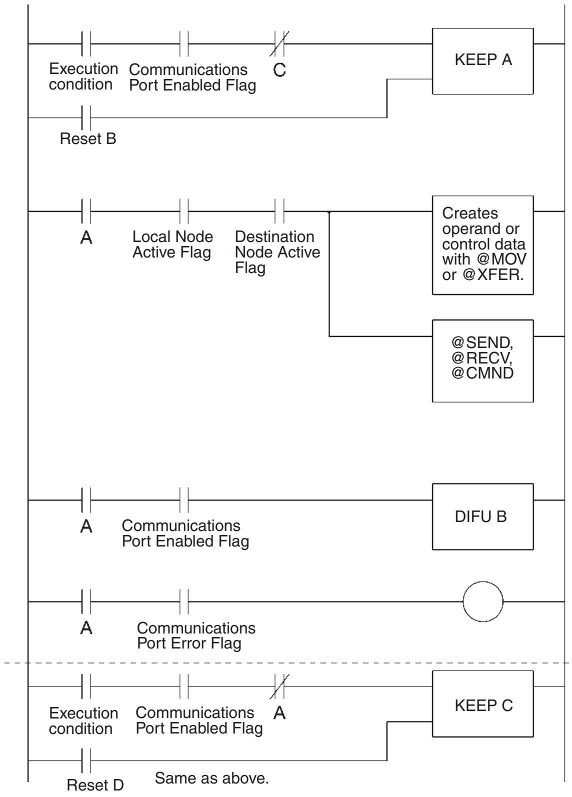
Address	Bits	Name	Description
A202	00 to 07	Communications Port Enabled Flags	Turns ON when a communications instruction can be executed. Bits 00 to 07 correspond to ports 0 to 7. You can tell when communications have been completed because the Communications Port Enabled Flag will turn ON again after the completing execution of the communications instruction. The Communications Port Error Flag will be OFF while the communications instruction is being executed.
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	Turns ON when a communications error occurs during execution of network communications. If the Communications Port Error Flag turns ON, refer to the Communications Port Completion Code in (A203 to A210) to identify the cause of the error. Bits 00 to 07 correspond to communications ports 0 to 7. The Communications Port Error Flag will be OFF while the communications instruction is being executed.

● Flag/Word Operation



Application Methods

1. Set the communications port in the instruction operand to 0 to 7 hex.
2. Use the Communications Port Enabled Flag (A202.00 to A202.07) to perform exclusive control of the communications port.



Bit A remains ON while the communications instruction is being executed.

Copies the operand and control data to the desired data area.

Executes the communications instruction.

Writes the reset input. (Reset B is turned ON when the communications instruction is completed.)

For Send Error Flag display (OK if data is retransmitted.)

↑
↓

Exclusive control to prevent simultaneous execution:
Exclusive control prevents another communications instruction from being executed until the instruction above is completed.

Automatic Allocation of Communications Ports

● Overview

Each of the following instructions use one of the communications ports (0 to 7) to perform network communications or serial communications.

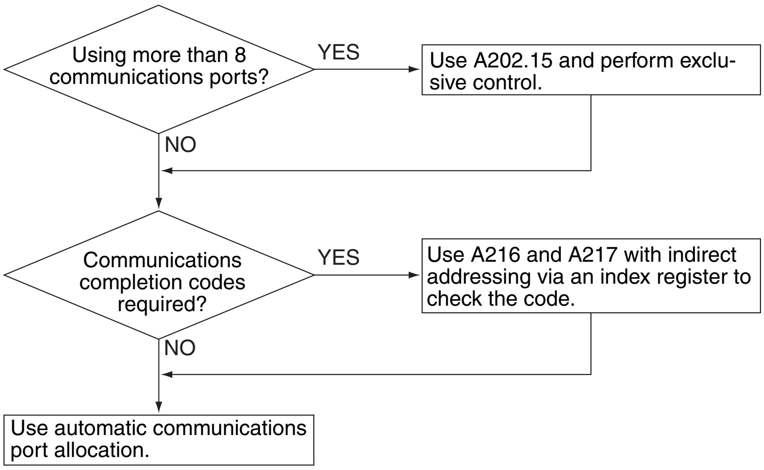
- Network Communications Instructions (SEND, RECV, and CMND)
- Serial Communications Instructions (PMCR, TXDU, and RXDU)
  The above instructions are collectively referred to as communications instructions.

Note Here, “communications port” indicates an internal logical port, not a physical port.

● Auxiliary Area Bits and Words Used When Automatically Allocating Communications Ports

Address	Bits	Name	Description
A202	15	Communications Port Allocation Enabled Flag	Turns ON when there is a communications port available for automatic allocation. **Note** Use this flag 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. **Note** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle. **Note** Use the Used Communications Port Number stored in A218 to determine which flag to access.
	08 to 15	Do not use.	---
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. If the flag turns ON, refer to the Communications Port Completion Code in (A203 to A210) to identify the cause of the error. **Note** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle. **Note** Use the Used Communications Port Number stored in A218 to determine which flag to access.
	08 to 15	Do not use.	---
A216 and A217	---	Communications Port Completion Code Storage Address	The completion code for a communications instruction is automatically stored at the address with the PLC I/O memory address given in these words. **Note** 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 Numbers	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.

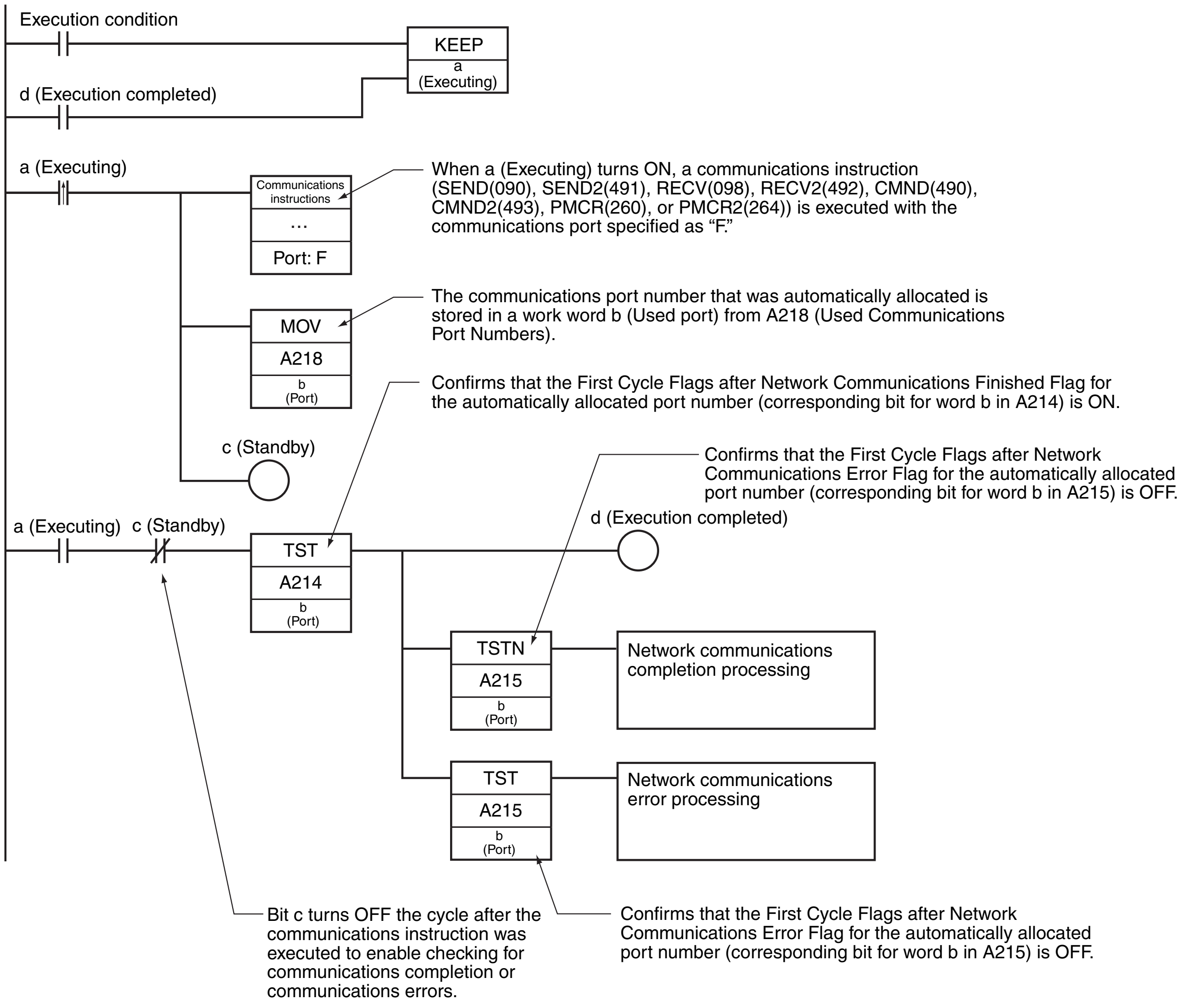
Use the following flowchart to determine whether to use the Communications Port Allocation Enabled Flag (A202.15) and the Communications Port Completion Code Storage Address (A216 and A217).



Applications Methods

1. Set the communications port in the instruction operand to F hex.
2. Program the ladder diagram as shown below.

● Completing and Processing Errors after Executing Communications Instructions



When a (Executing) turns ON, a communications instruction (SEND(090), SEND2(491), RECV(098), RECV2(492), CMND(490), CMND2(493), PMCR(260), or PMCR2(264)) is executed with the communications port specified as "F."

The communications port number that was automatically allocated is stored in a work word b (Used port) from A218 (Used Communications Port Numbers).

Confirms that the First Cycle Flags after Network Communications Finished Flag for the automatically allocated port number (corresponding bit for word b in A214) is ON.

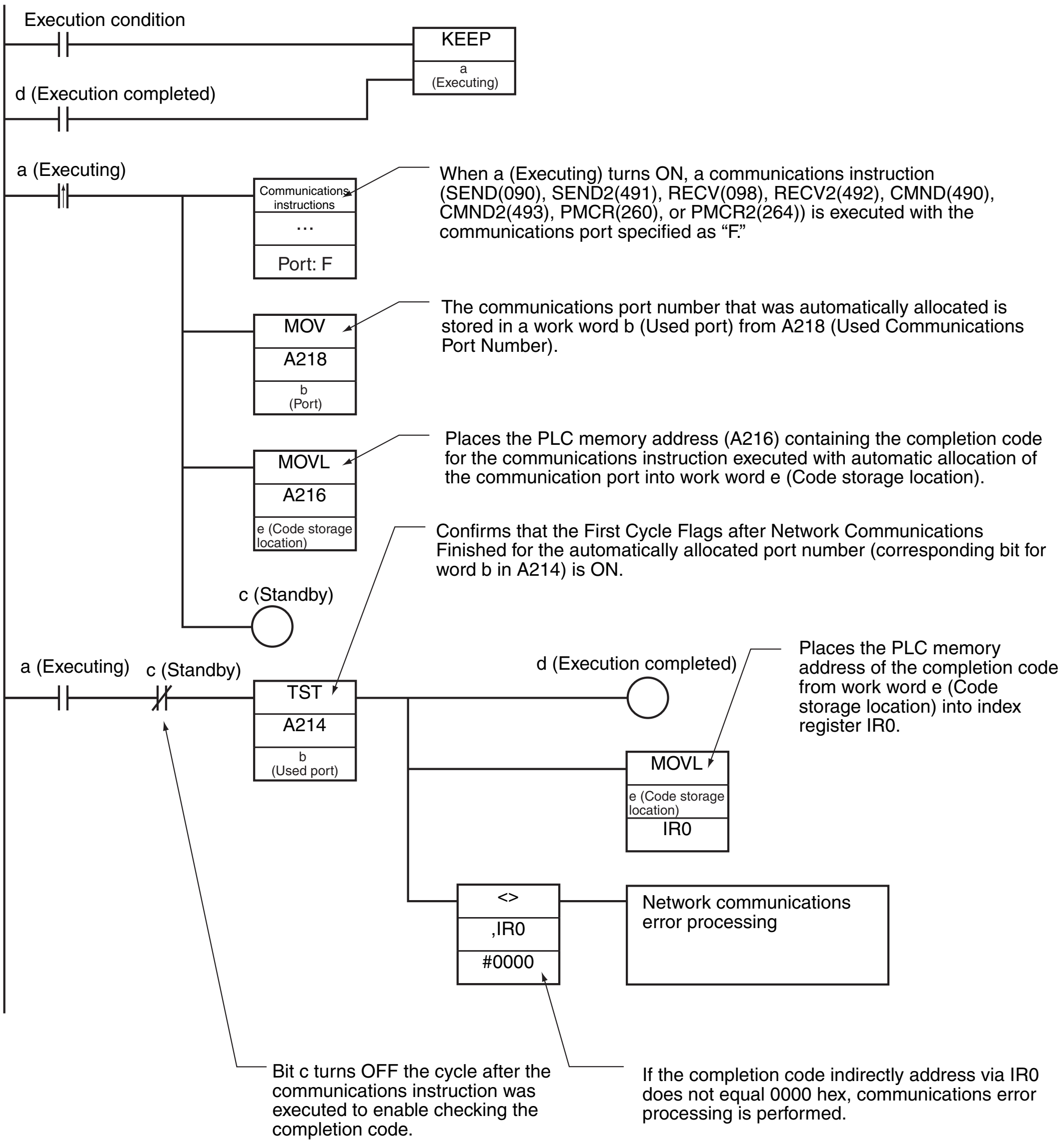
Confirms that the First Cycle Flags after Network Communications Error Flag for the automatically allocated port number (corresponding bit for word b in A215) is OFF.

Bit c turns OFF the cycle after the communications instruction was executed to enable checking for communications completion or communications errors.

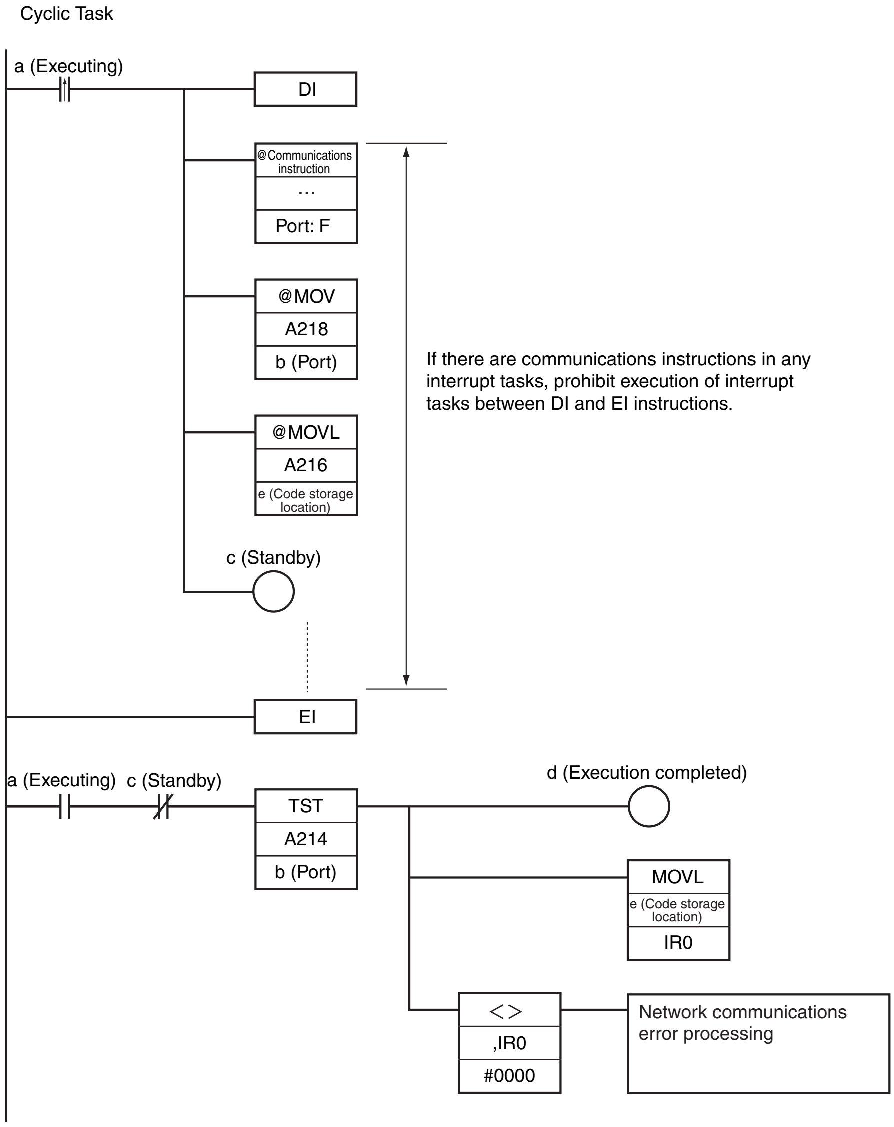
Confirms that the First Cycle Flags after Network Communications Error Flag for the automatically allocated port number (corresponding bit for word b in A215) is OFF.

Accessing the Completion Code after Executing Communications Instructions

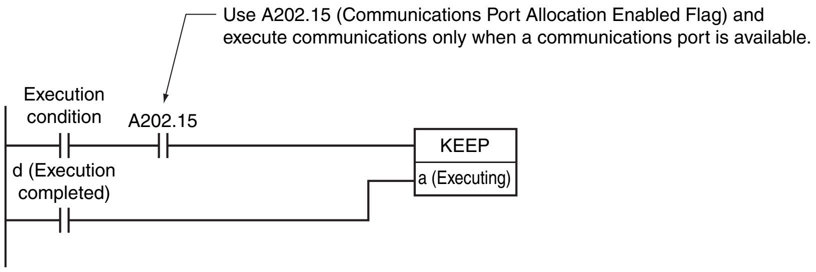
The completion codes are generally used to troubleshoot errors when they occur. A completion code of 0000 hex can, however, also be used to confirm that communications have been completed normally.



Note 1 If you use communications instructions inside interrupt tasks (regardless of whether you specify communications ports or use automatic port allocation), use the DI and EI instructions to prohibit executing interrupt tasks when executing communications instructions with automatic port allocation in cyclic tasks, as shown below.

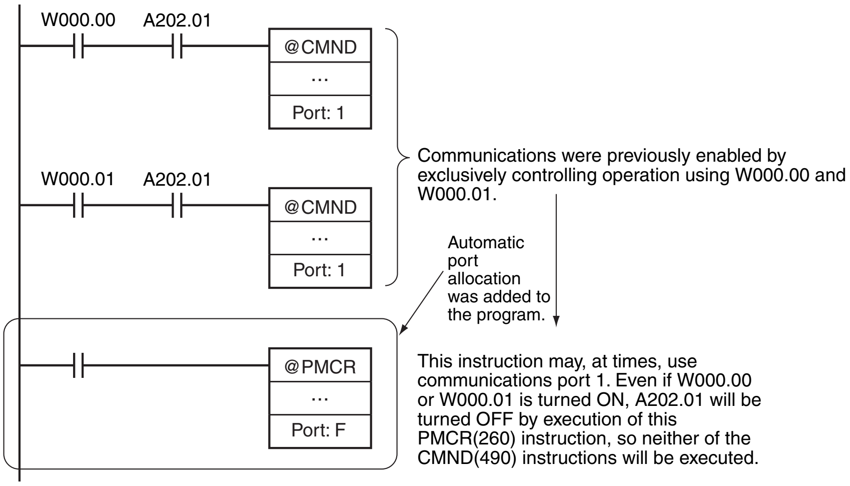


2 If it is possible that more than eight communications instructions will be executed at the same time, always check to be sure there is an available communications port before executing a communications instruction, even when using automatic port allocation.



3 It is acceptable to specify the communications port for some communications instructions and use automatic port allocation for others. It is possible, however, that a port specified by the user for one communications instructions may have already been automatically allocated. You must therefore be careful when adding communications instructions that use automatic port allocation to existing ladder diagrams, as shown below.

Programming Example



Specifications of CJ2 Network Communications Instructions (SEND2, CMND2, PMCR2, and RECV2)

● Communications Port Allocations

Although the communications port (see note) was specified in the operands of previous Network Communications Instructions, the CJ2 Network Communications Instructions use automatic port allocation with new communications ports (8 to 71) to enable executing up to 64 of the CJ2 Network Communications Instruction simultaneously. With these instructions, there is no need for the user to keep track of the communications ports.

Each time one of these instructions is executed, the number of ports stored in A211 (Number of Ports Available) will be decremented. Each time a response is returned for a communications port, the value in A211 will be incremented. This system enables monitoring communications performance.

Communications ports	Application	Assignment method
0 to 7	Network Communications Instructions, Serial Communications Instructions, easy backup (SEND, CMND, PMCR, TXDU, etc.)	Specified in operand. 0 to 7: The specified communications port is assigned. F: The communications port is automatically assigned between 0 and 7.
8 to 71	CJ2 Network Communications Instructions (SEND2, CMND2, PMCR2, and RECV2)	Automatically assigned, with the number of available ports stored in A211.

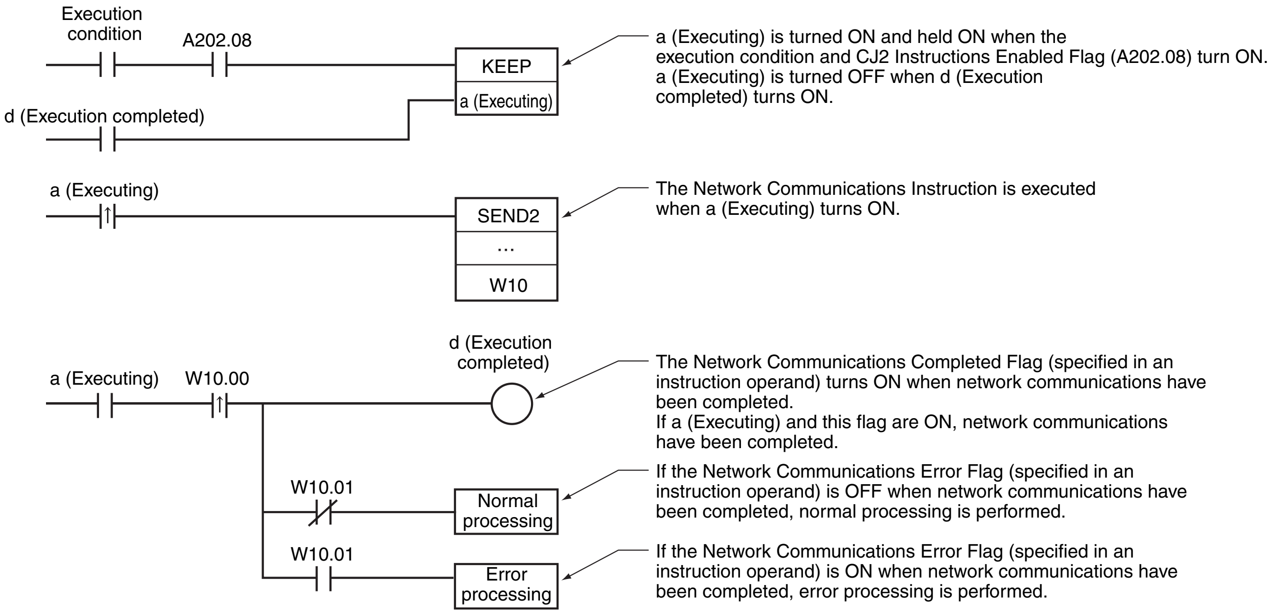
Note The communications ports are internal logical ports (virtual ports) used to manage simultaneous execution of communications instructions over multiple cycles (including communications processing, file handling, etc.).

● Auxiliary Area Words and Bits

The CJ2 Network Communications Instructions use the communications information in operand I in place of the Communications Port Enabled Flags in A202 used by previous instructions.

Auxiliary Area	Network Communications Instructions for CJ1/CS1/CJ2	CJ2 Network Communications Instructions
A202.00 to A202.07	Communications Port Enabled Flag (communications ports 0 to 7)	Instruction operand I Bit 0 (Turns ON when execution is completed, not when execution is enabled.)
A202.08	Not used.	CJ2 Instructions Enabled Flag (default: ON)
A203 to 210	Communications completion codes for communications ports 0 to 7	Instruction operand I+1
A211	Not used.	Number of ports available for CJ2 Network Communications Instruction (A202.08 turns OFF when the value in A211 reaches 0.)
A213	Explicit Communications Error Flag	
A214.00 to A214.07	First Cycle Flags after Network Communications Finished	---
A215.00 to A215.07	First Cycle Flags after Network Communications Error	---
A216 and A217	Communications Port Completion Code Storage Address	---
A218	Used Communications Port Numbers	---
A219	Communications Port Error Flags	Instruction operand I Bit 1

Applications Methods



a (Executing) is turned ON and held ON when the execution condition and CJ2 Instructions Enabled Flag (A202.08) turn ON. a (Executing) is turned OFF when d (Execution completed) turns ON.

The Network Communications Instruction is executed when a (Executing) turns ON.

The Network Communications Completed Flag (specified in an instruction operand) turns ON when network communications have been completed.
If a (Executing) and this flag are ON, network communications have been completed.

If the Network Communications Error Flag (specified in an instruction operand) is OFF when network communications have been completed, normal processing is performed.

If the Network Communications Error Flag (specified in an instruction operand) is ON when network communications have been completed, error processing is performed.

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) or CMND2(493) to send a FINS command code of 2801 hex (EXPLICIT MESSAGE SEND).
- Use the following Explicit Message Instructions. (See note.)

Note These instructions are supported only by CS/CJ-series CPU Unit with unit version 2.0 or later.

Explicit Message Instructions

The following instructions, which are used specially for explicit messages, are called Explicit Message Instructions.

Mnemonic	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) or CMND2(493) 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 (722)	EXPLICIT SET ATTRIBUTE	Sends an explicit message with a service code of 10 hex (SET ATTRIBUTE SINGLE).
ECHRD (723)	EXPLICIT WORD READ	Uses an explicit message to read data from a CPU Unit.
ECHWR (724)	EXPLICIT WORD WRITE	Uses an explicit message to write data to a CPU Unit.

Features of Explicit Message Instructions

- This enables easily reading and writing data between CPU Units using explicit message communications.
- 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) or CMND2(493) instructions with service code 1E hex (word data read) or 1F hex (word data write).)

Operation

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).

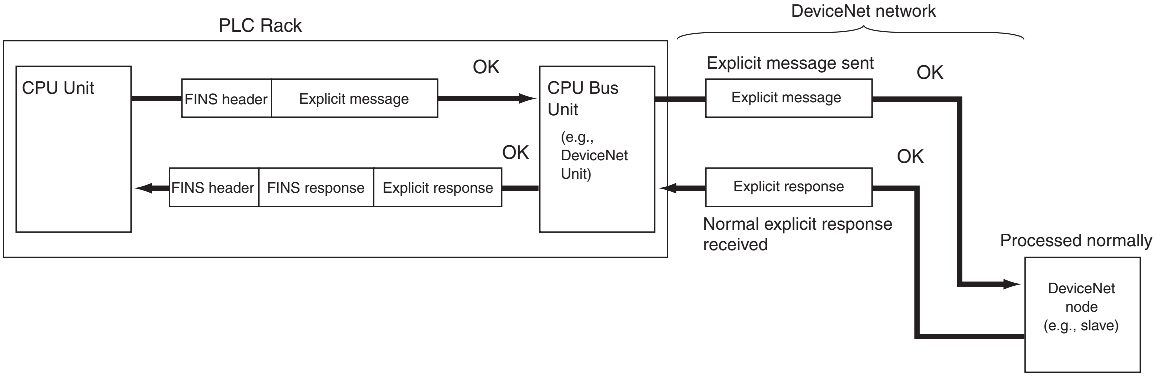
The Communications Port Completion Code will contain 0000 hex after a normal end, an explicit message error code after an explicit communications error end, and a FINS message completion code after a FINS error end.

Condition		Explicit Communications Error Flag (A213.00 to A213.07: Communications ports 0 to 7)	Communications Port Error Flag (A219.00 to A219.07: Communications ports 0 to 7)	Communications Port Completion Code (A203 to A210: Communications ports 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

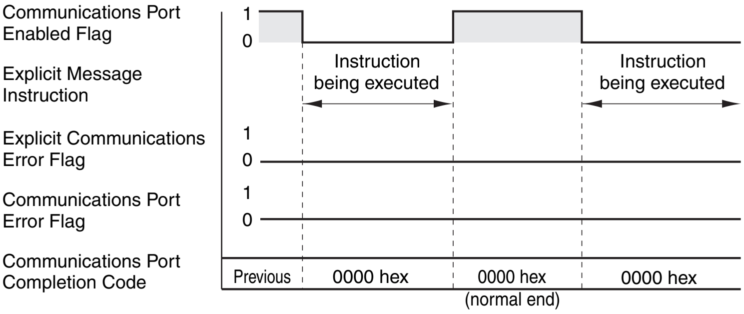
● Normal End

An explicit message is sent and a normal response is returned.

The corresponding Explicit Communications Error Flag (A213.00 to A213.07: Communications ports 0 to 7) will be OFF and the Communications Port Completion Code (A203 to A210: Communications ports 0 to 7) will contain the explicit message normal completion code of 0000 hex.



Communications Flags



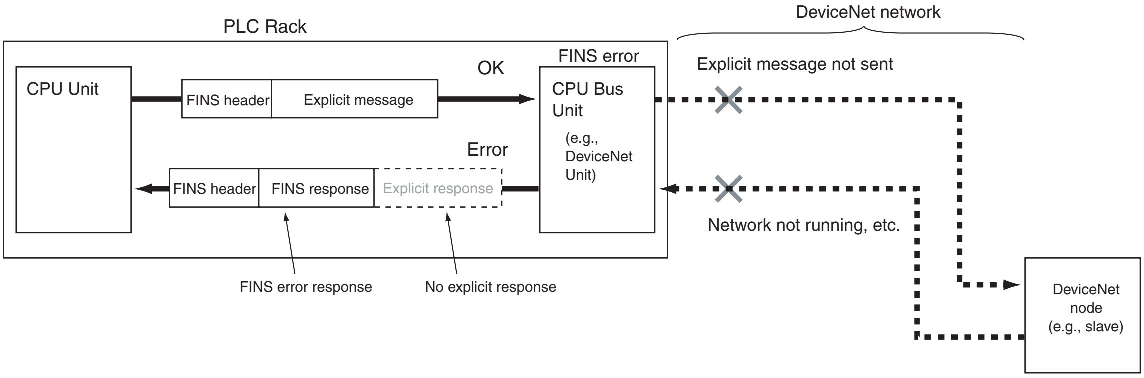
● Error End

There are two possibilities for error ends, as described in the next two subsections.

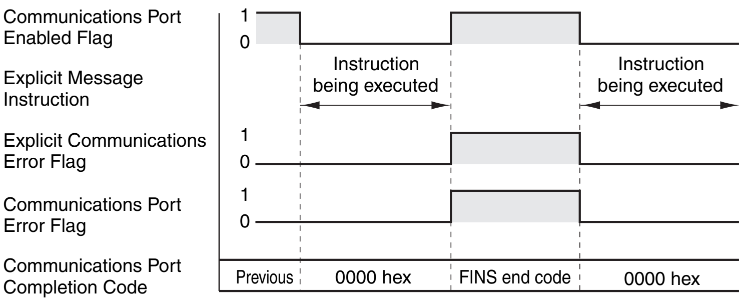
a) When the Explicit Message Could Not Be Sent

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 ports 0 to 7) and the Communications Port Error Flag (A219.00 to A219.07: Communications ports 0 to 7) will turn ON.

After completion, the Communications Port Completion Code (A203 to A210: Communications ports 0 to 7) will contain the FINS message error code.



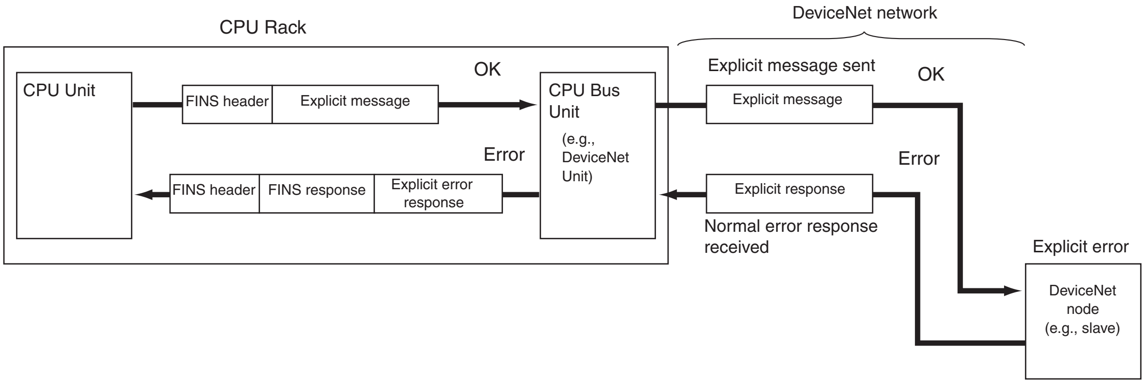
Communications Flags



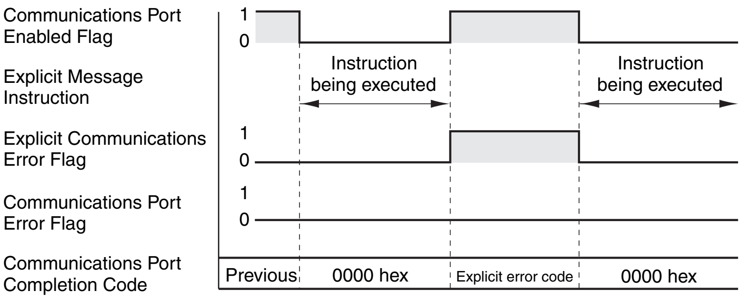
b) When the Explicit Message Was Sent But an Explicit Error Response Was Returned

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 ports 0 to 7) will turn ON and the Communications Error Flag (A219.00 to A219.07: Communications ports 0 to 7) will remain OFF.

After completion, the Communications Port Completion Code (A203 to A210: Communications ports 0 to 7) will contain the explicit message error code.

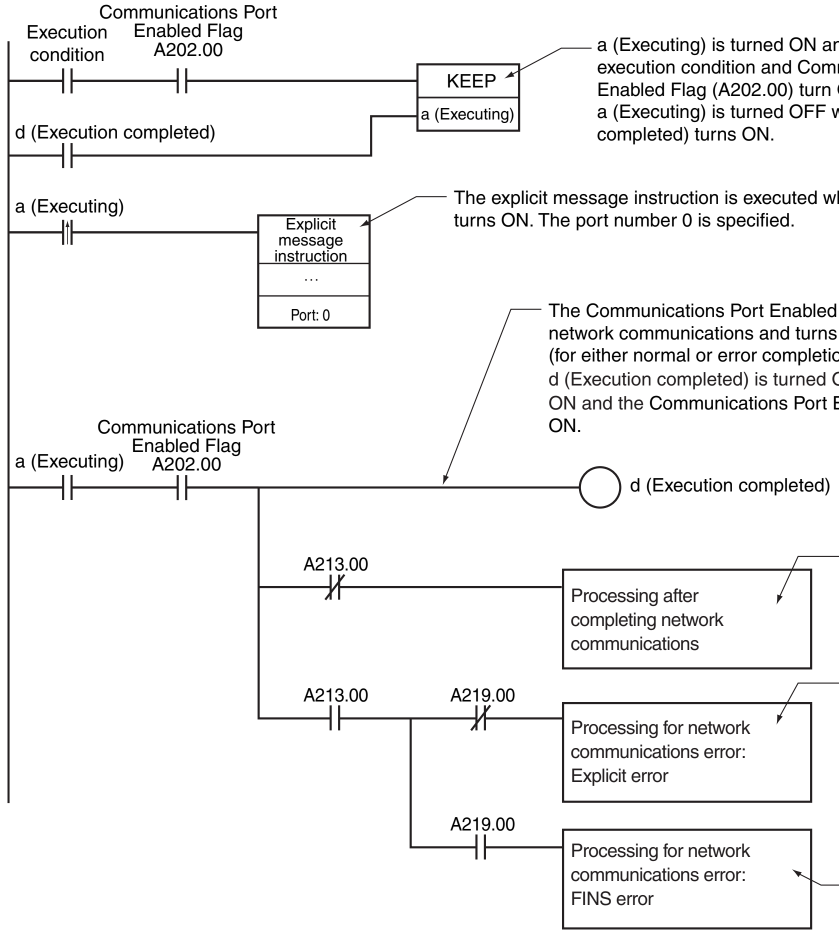


Communications Flags



Ladder Programming Examples

● User Specification of Communications Ports



a (Executing) is turned ON and held ON when the execution condition and Communications Port Enabled Flag (A202.00) turn ON.
a (Executing) is turned OFF when d (Execution completed) turns ON.

The explicit message instruction is executed when a (executing) turns ON. The port number 0 is specified.

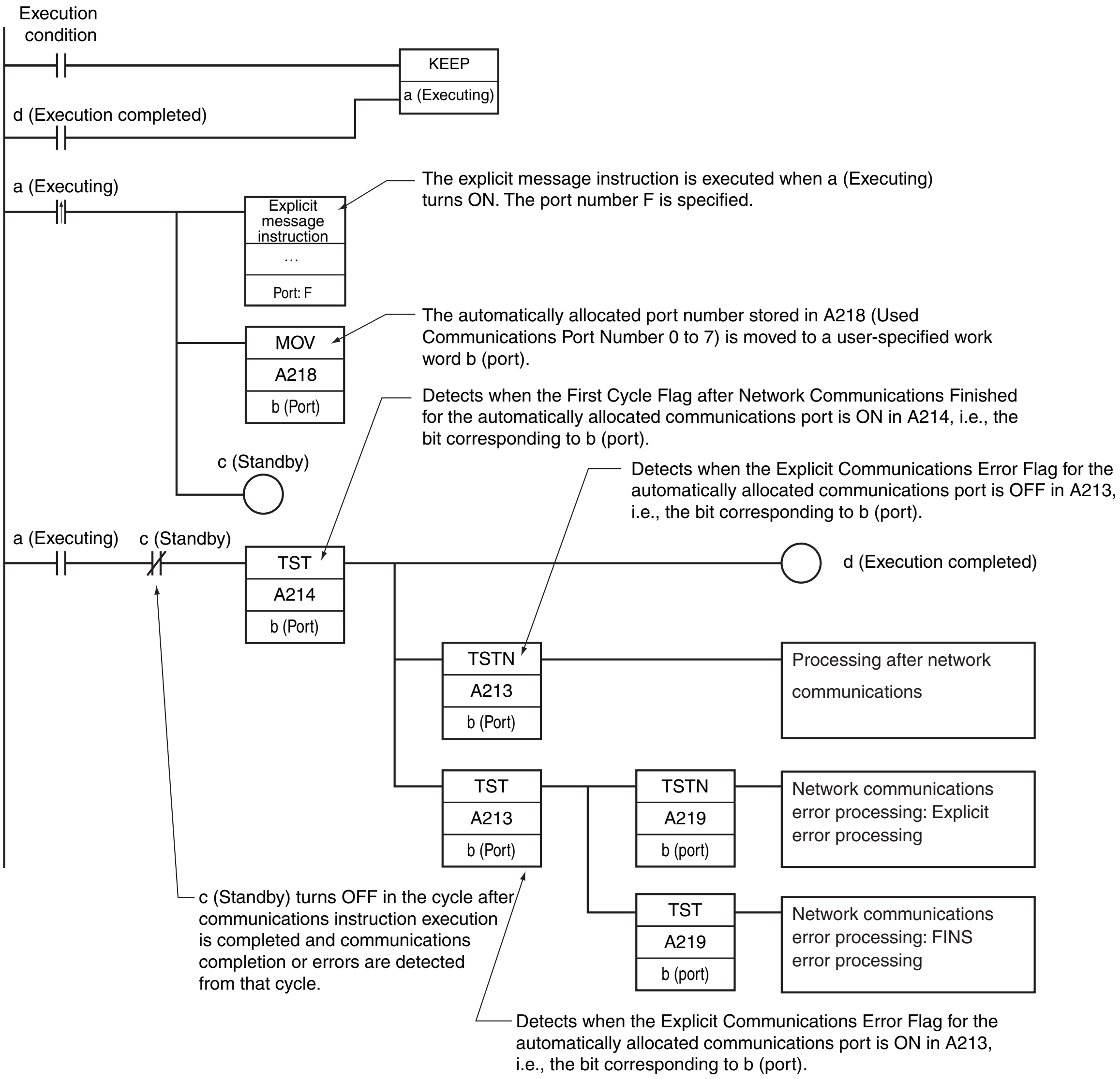
The Communications Port Enabled Flag (A202.00) is OFF during network communications and turns ON when they are completed (for either normal or error completion).
d (Execution completed) is turned ON when a (Executing) turns ON and the Communications Port Enabled Flag (A202.00) turns ON.

If the Explicit Communications Error Flag (A213.00) when execution is completed, explicit memory communications were completed normally and normal processing after network communications is performed.

If the Explicit Communications Error Flag (A213.00) is ON when execution is completed, an error has occurred in explicit message communications and the Communications Port Error Flag (A219.00) is checked. If it is OFF, processing for an explicit communications error is performed on the assumption that an explicit message was sent and an explicit message response was received.

If the Explicit Communications Error Flag (A213.00) is ON when execution is completed, an error has occurred in explicit message communications and the Communications Port Error Flag (A219.00) is checked. If it is ON, processing for a FINS communications error is performed on the assumption that an explicit message was never sent.

● Automatically Allocating Communications Ports (i.e., Internal Logical Ports)



The explicit message instruction is executed when a (Executing) turns ON. The port number F is specified.

The automatically allocated port number stored in A218 (Used Communications Port Number 0 to 7) is moved to a user-specified work word b (port).

Detects when the First Cycle Flag after Network Communications Finished for the automatically allocated communications port is ON in A214, i.e., the bit corresponding to b (port).

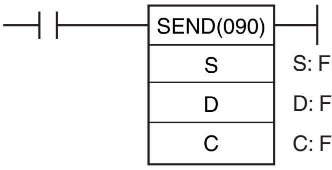
Detects when the Explicit Communications Error Flag for the automatically allocated communications port is OFF in A213, i.e., the bit corresponding to b (port).

c (Standby) turns OFF in the cycle after communications instruction execution is completed and communications completion or errors are detected from that cycle.

Detects when the Explicit Communications Error Flag for the automatically allocated communications port is ON in A213, i.e., the bit corresponding to b (port).

SEND

Instruction	Mnemonic	Variations	Function code	Function
NETWORK SEND	SEND	@SEND	090	Sends data to a node in the network.

	SEND
Symbol	 S: First source word (local node) D: First destination word (remote node) C: First control word

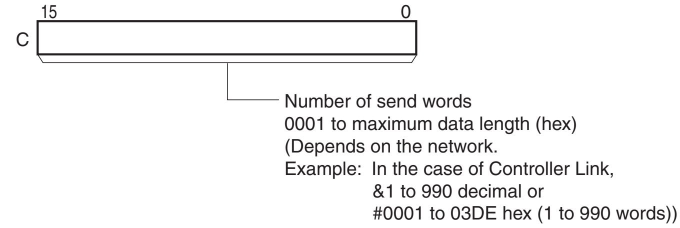
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

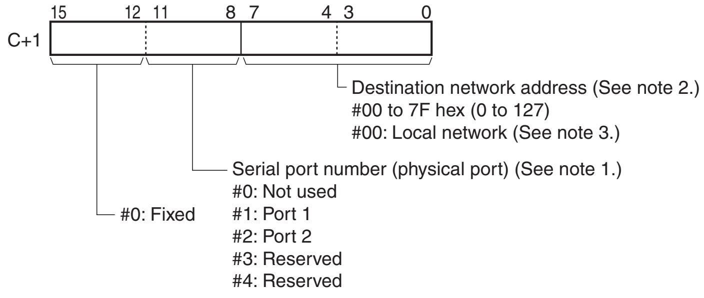
Operands

Operand	Description	Data type	Size
S	First source word (local node)	UINT	Variable
D	First destination word (remote node)	UINT	Variable
C	First control word	WORD	5

C: First Control Word



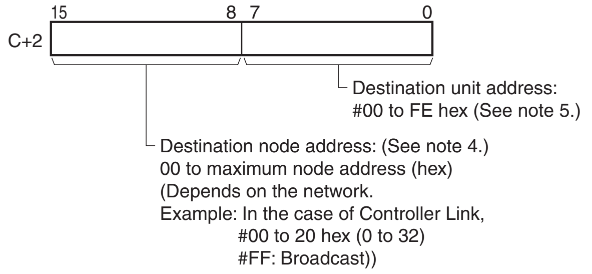
Number of send words
0001 to maximum data length (hex)
(Depends on the network.
Example: In the case of Controller Link,
&1 to 990 decimal or
#0001 to 03DE hex (1 to 990 words))



Destination network address (See note 2.)
#00 to 7F hex (0 to 127)
#00: Local network (See note 3.)

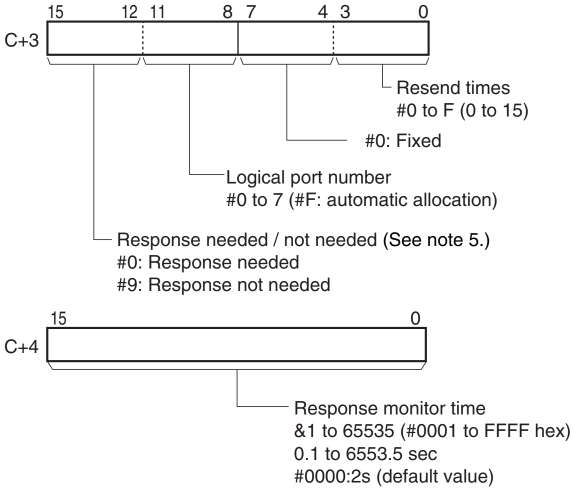
Serial port number (physical port) (See note 1.)
#0: Not used
#1: Port 1
#2: Port 2
#3: Reserved
#4: Reserved

#0: Fixed



Destination unit address:
#00 to FE hex (See note 5.)

Destination node address: (See note 4.)
00 to maximum node address (hex)
(Depends on the network.
Example: In the case of Controller Link,
#00 to 20 hex (0 to 32)
#FF: Broadcast))



Note 1 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.

1) Set the destination unit address (bits 00 to 07 of C+2) to the unit address of the CPU Unit or Serial Communications Unit/Board 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	CPU Unit	1 hex	Built-in RS-232C port
		2 hex	Peripheral port
10 hex + unit number	Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1
		2 hex	Port 2
E1 hex	Serial Communications Board (Inner Board) (CS Series only)	1 hex	Port 1
		2 hex	Port 2

2) 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.

- 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)

- Serial Communication Board ports

Port	Port’s unit address
Port 1	E4 hex (228 decimal)
Port 2	E5 hex (229 decimal)

- CPU Unit ports

Port	Port’s unit address
Peripheral	FD hex (253 decimal)
RS-232C	FC hex (252 decimal)

2 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.

3 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.

4 For a broadcast transmission, set #FF hex.
To send within the same node, set #00 hex.

5 The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
CPU Unit	00 hex
CPU Bus Unit	10 hex + unit number
Special I/O Unit (except C200H-series Special I/O Units)	20 hex + unit number
Inner Board (CS Series only)	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	Serial Communications Unit ports Port 1: 80 hex + 4 × unit number Port 2: 81 hex + 4 × unit number Serial Communications Board ports Port 1: E4 hex (228 decimal) Port 2: E5 hex (229 decimal) CPU Unit ports Peripheral port: FD hex (253 decimal) RS-232C port: FC hex (252 decimal)

6 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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK*	OK	OK	---	---	---	OK	---	---	---	---
D																		
C																		

* The D bank and higher of the EM Area can be specified only for CJ2 CPU Units, or CS1D-CPU□□HA. The D bank and higher cannot be specified in the program in a CS1/CJ1-series CPU Unit even if the destination node is a CJ2 CPU Unit or CS1D-CPU□□HA.

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. • ON if SEND(090) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

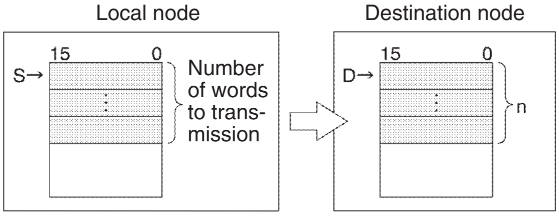
Related Auxiliary Area Words and Bits

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.

Note Refer to the FINS command response codes in the *CS/CJ Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Function

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.

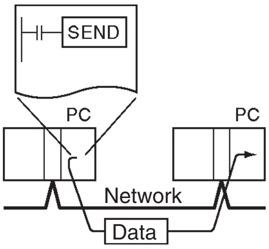


- If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when SEND(090) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A202.00 to A202.07) and Communications Port Error Flag (ports 00 to 07: A219.00 to A219.07) will be turned OFF and 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.
- 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.)
- 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.

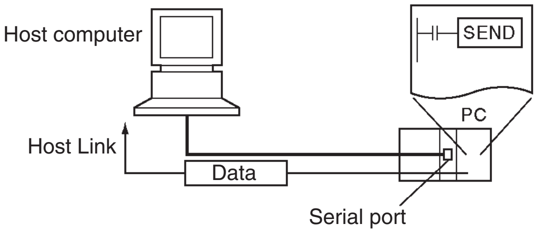
● Transmission through the Network

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.



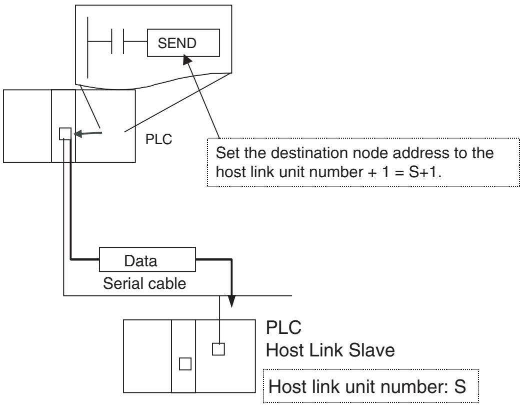
● Transmission through Host Link

When the CPU Unit's built-in serial port, a Serial Communications Board (CS-series only), or 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

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.



Hint

● Sending data to a host computer connected by Host Link (non-solicited communications)

If SEND(090) is sent to the serial port of the CPU Unit, a Serial Communications Board (CS Series only), or 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.

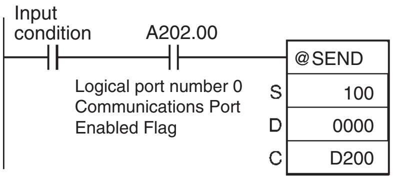
A program must be created in the host computer to process the received command (the FINS command enclosed in the host link header and terminator).

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 (CPU Unit), E1 (Inner Board (CS Series only), or unit number + 10 hexadecimal (Serial Port Unit).

- SEND instruction (sending data to a host computer):

When the input condition and A202.00 (the Communications Port Enabled Flag for port 0) 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 Serial Communications Unit with unit address 10 (hex) at node number 3 in network 0.

Note It is necessary create a program at the host computer to receive the data and send a response.



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 Board
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)

Precaution

- 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.
- Refer to *Automatically Allocating Communications Ports (i.e., Internal Logical Ports)* in *SECTION 3 Instructions* - *Network Instructions* for details on using automatic allocation of the communications port number (logical port).
- 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.

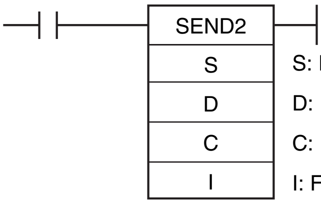
Example Programming

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)

SEND2

Instruction	Mnemonic	Variations	Function code	Function
NETWORK SEND 2	SEND2	@SEND2	491	Sends data to a node on a network.

	SEND2
Symbol	 S: First source word D: First destination word C: First control word I: First communications information word

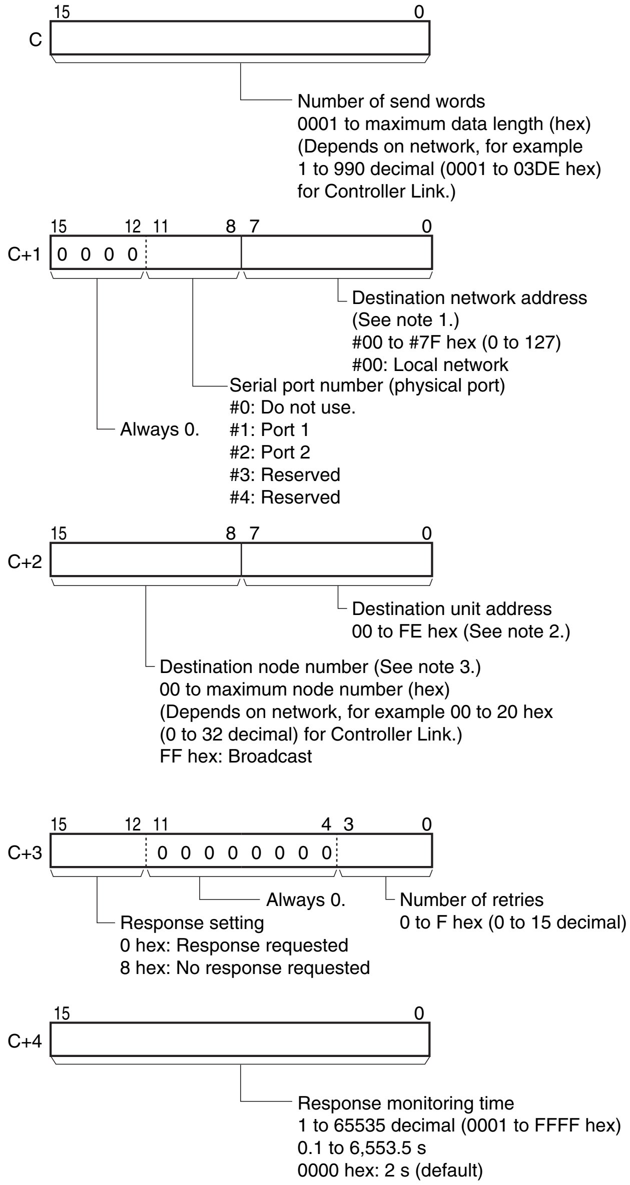
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

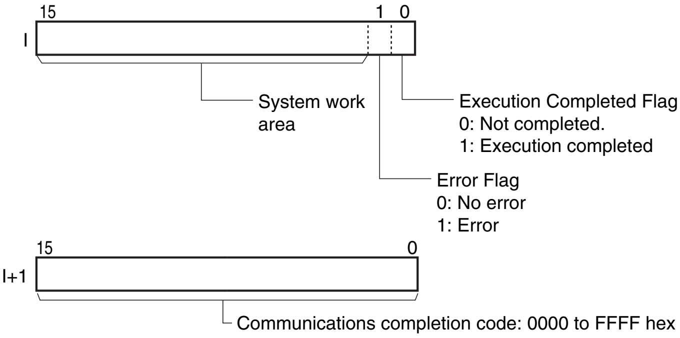
Operand	Description	Data type	Size
S	First source word	WORD	Variable
D	First destination word	WORD	Variable
C	First control word	WORD	6
I	First communications information word	WORD	2

C: First Control Word



I: First communications information word

Clear D and I words before executing the instruction.



Note 1 Set the destination network address in the routing tables unless you are specifying a local network. Set the destination network address to 00 for a local network. If there is more than one CPU Bus Unit mounted in the PLC, the “local network” is the network of the CPU Bus Unit with the lowest unit number.

2 The unit addresses are as follows:

- CPU Unit: 00 hex
- CPU Bus Unit: 10 hex + unit number (hex)
- Special I/O Unit: 20 hex + unit number (hex) (Except for C200H Special I/O Units)
- Inner Board: E1 hex (CS Series only)
- Computer: 01 hex
- Unit connected to network: FE hex (Eliminates the need to specify the unit number of the remote Unit.)

The serial port unit address are as follows:

- Serial Communications Units:
  Port 1: 80 hex + 04 hex × unit number (hex)
  Port 2: 81 hex + 04 hex × unit number (hex)
- Serial Communications Boards:
  Port 1: E4 hex (228 decimal)
  Port 2: E5 hex (229 decimal)
- CPU Unit
  Peripheral port: FD hex (253 decimal)
  RS-232C port: FC hex (252 decimal)

3 Set FF hex to broadcast. Set 00 hex to send to the local node.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D																		
C												---						
I																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the serial port number specified in C+1 is not between 0 and 4 hex. • ON if there is no communications port available (i.e., if A211 = 0). • OFF in all other cases.

Function

When SEND2(491) is executed, the specified number of send words starting with the first source word specified in S at the source node are sent to the destination node (i.e., the remote node) in the destination network specified in C. The data is written starting from the destination word address specified in D1 in the node with the specified destination unit address.

When the transmission starts, the Execution Completed Flag in I is turned OFF. When a response is returned, the transmission results is shown in the Error Flag in I. The communications response is written to I+1.

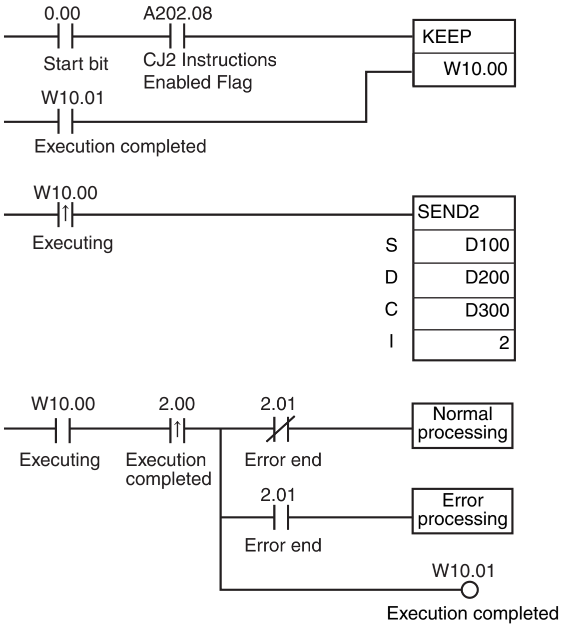
The Communications Port Enabled Flags (A202.00 to A202.07) and Communications Port Error Flags (A219.00 to A219.07) used for the SEND(090) instruction are not used for SEND2(491), which uses the operand I. The communications port completion code (previously stored in A203 to A210) is stored in I+1.

Example Programming

If CIO 0.00 (start bit) and A202.08 (CJ2 Instructions Enabled Flag) are ON, W10.00 (executing) will turn ON.

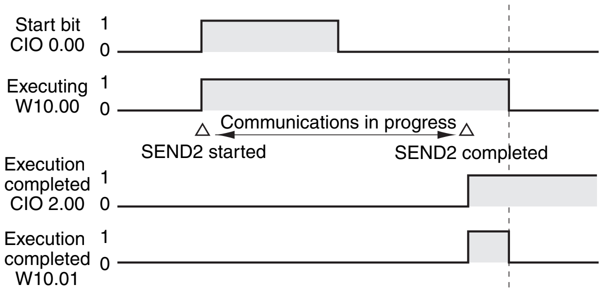
When W10.00 (Executing) turns ON, the ten words of data starting from D100 (D100 to D109) are sent to the CPU Unit with node number 3 in the local network where they are written to ten words starting from D200 (D200 to D209). If a response is not received within 10 seconds, the data will be resent up to 3 times.

If communications end normally, CIO 2.00 (execution completed) turns ON, and normal processing is executed. If communications end in an error, CIO 2.01 (error end) turns ON, and error processing is executed.



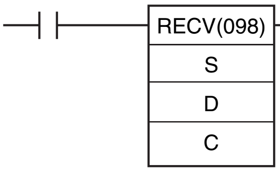
C: D300	0 0 1 0	Number or words to send: 10 words
C+1: D301	0 0 0 0	Transmit to the local network and the device itself
C+2: D302	0 3 0 0	Node number 03, unit address 00 (CPU Unit)
C+3: D303	0 0 0 3	Response requested, 3 retries
C+4: D304	0 1 0 0	Response monitoring time: 0064 hexadecimal (10 seconds)

● Timing Chart



RECV

Instruction	Mnemonic	Variations	Function code	Function
NETWORK RECEIVE	RECV	@RECV	098	Requests data to be transmitted from a node in the network and receives the data.

	RECV
Symbol	 S: First source word (remote node) D: First destination word (local node) C: First control word

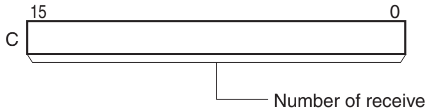
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

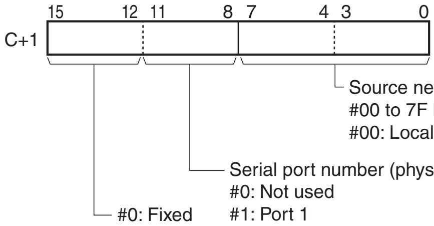
Operands

Operand	Description	Data type	Size
S	First source word (remote node)	UINT	Variable
D	First destination word (local node)	UINT	Variable
C	First control word	WORD	5

C: First Control Word



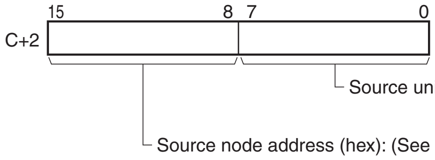
Number of receive (send request) words
0001 to maximum data length (hex)
(Depends on the network.
Example: In the case of Controller Link,
&1 to 990 decimal or
#0001 to 03DE hex)



Source network address (See note 2)
#00 to 7F hex (0 to 127)
#00: Local network (See note 3)

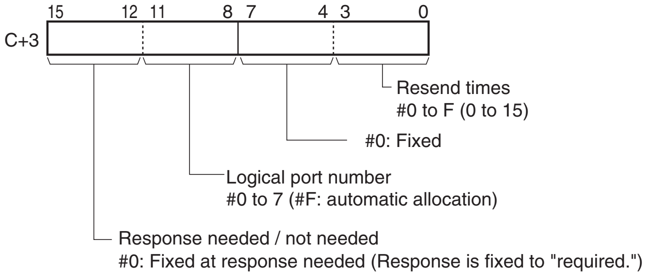
Serial port number (physical port) (See note 1)
#0: Not used
#1: Port 1
#2: Port 2

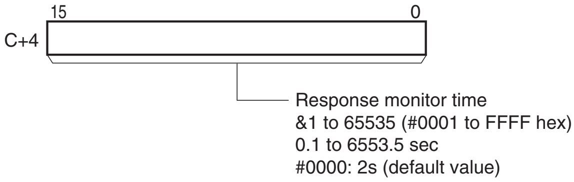
#0: Fixed



Source unit address (See note 5)

Source node address (hex): (See note 4)
00 to maximum node address (hex)
(Depends on the network.
Example: In the case of Controller Link,
#00 to 20 hex (0 to 32))





Note 1 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.

1) Set the source unit address (bits 00 to 07 of C+2) to the unit address of the CPU Unit or Serial Communications Unit/Board 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	CPU Unit	1 hex	Built-in RS-232C port
		2 hex	Peripheral port
10 hex + unit number	Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1
		2 hex	Port 2
E1 hex	Serial Communications Board (Inner Board) (CS Series only)	1 hex	Port 1
		2 hex	Port 2

2) 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.

- 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)

- Serial Communication Board ports

Port	Port’s unit address
Port 1	E4 hex (228 decimal)
Port 2	E5 hex (229 decimal)

- CPU Unit ports

Port	Port’s unit address
Peripheral	FD hex (253 decimal)
RS-232C	FC hex (252 decimal)

2 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.

3 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.

4 Broadcast transmission is not possible using a RECV(098) instruction. For a send request within the same node, set #00.

5 Unit address

• CPU Unit:	00 hex
• CPU Bus Unit:	10 hex + unit number
• Special I/O Unit (except C200H-series Special I/O Units):	20 hex + unit number
• Inner Board (CS Series only):	E1 hex
• Computer:	01 hex
• Unit connected to network (not necessary to specify Unit):	FE hex

Serial Port's unit address

- Serial Communications Unit ports
  - Port 1: 80 hex + 4 × unit number
  - Port 2: 81 hex + 4 × unit number
- Serial Communications Board ports
  - Port 1: E4 hex (228 decimal)
  - Port 2: E5 hex (229 decimal)
- CPU Unit ports
  - Peripheral port: FD hex (253 decimal)
  - RS-232C port: FC hex (252 decimal)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK*	OK	OK	---	---	---	OK	---	---	---	---
D																		
C																		

* The D bank and higher of the EM Area can be specified only for CJ2 CPU Units, or CS1D-CPU□□HA. The D bank and higher cannot be specified in the program in a CS1/CJ1-series CPU Unit even if the destination node is a CJ2 CPU Unit or CS1D-CPU□□HA.

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. • ON if RECV(098) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

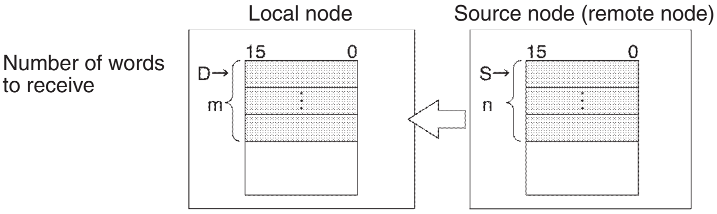
Related Auxiliary Area Words and Bits

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.

Note Refer to the FINS command response codes in the *CS/CJ Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Function

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.

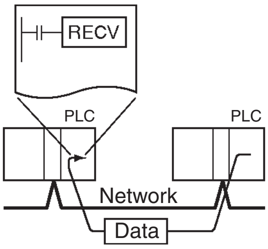


- If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when SEND(090) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A202.00 to A202.07) and Communications Port Error Flag (ports 00 to 07: A219.00 to A219.07) will be turned OFF and 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.
- 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.)
- 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.

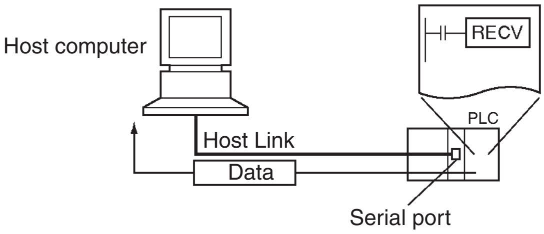
● Transmission through the Network

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.



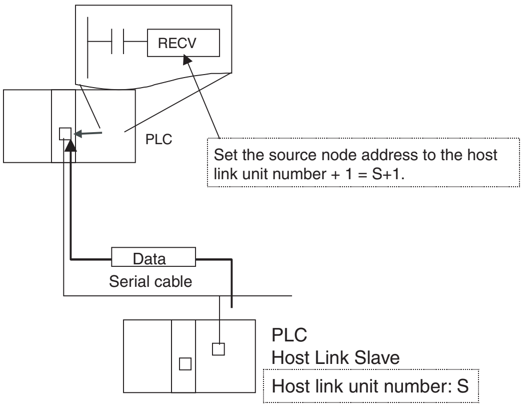
● Transmission through Host Link

When the CPU Unit's built-in serial port, a Serial Communications Board (CS Series only), or Serial Communications Unit is in host link mode 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.



● 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 Board or Unit. In this case, the source node address must be set to the host link unit number + 1.



Hint

● Transmission through Host Link

If RECV(098) is executed for the serial port of the CPU Unit, a Serial Communications Board (CS Series only), or 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).

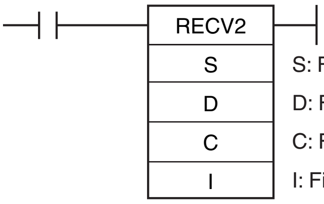
If the destination serial port is in the local PLC, set the network address to 00 (local network) in C+1, set the node address to 00 (local PLC) in C+2, and set the unit address to 00 (CPU Unit), E1 (Inner Board, CS Series only), or unit number + 10 hexadecimal (Serial Port Unit).

Precaution

- 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.
- Refer to *Automatically Allocating Communications Ports (i.e., Internal Logical Ports)* in *SECTION 3 Instructions* - *Network Instructions* for details on using automatic allocation of the communications port number (logical port).
- 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.

RECV2

Instruction	Mnemonic	Variations	Function code	Function
NETWORK RECEIVE 2	RECV2	@RECV2	492	Requests data to be transmitted from a node in the network and receives the data.

	RECV2
Symbol	 S: First source word D: First destination word C: First control word I: First communications information word

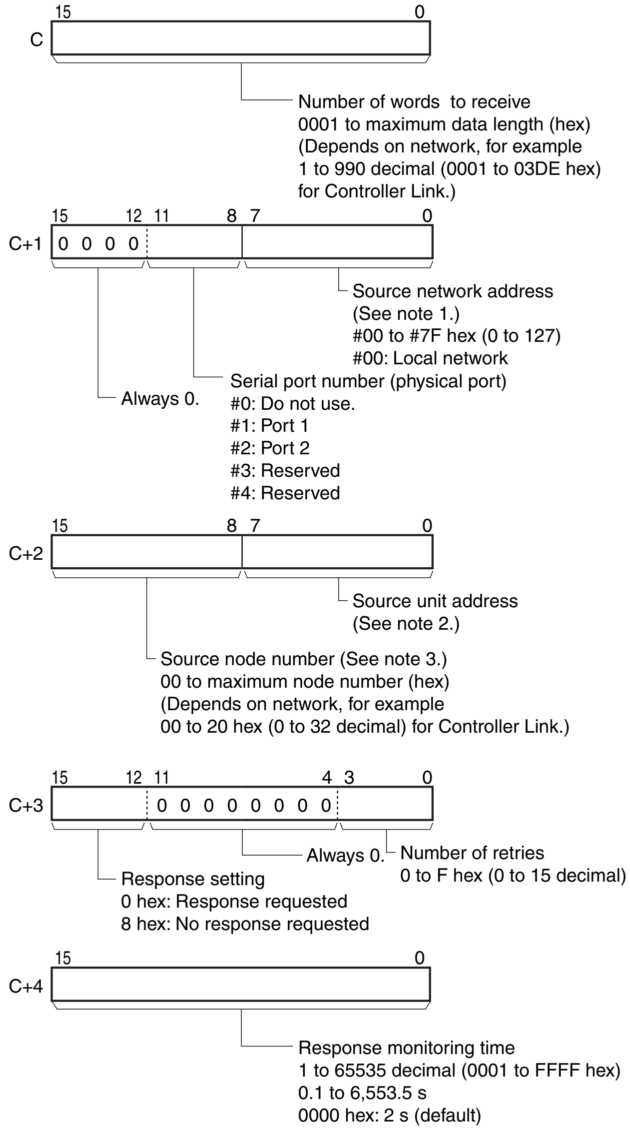
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

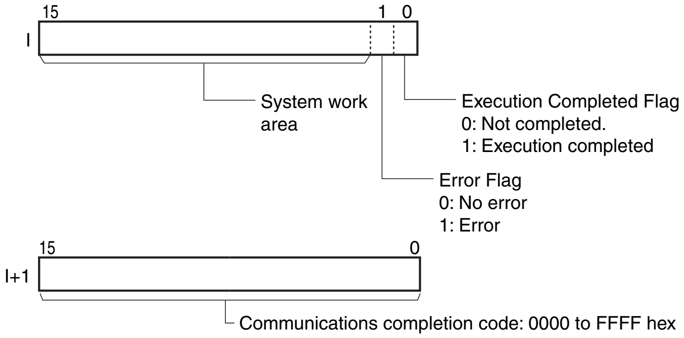
Operand	Description	Data type	Size
S	First source word	WORD	Variable
D	First destination word	WORD	Variable
C	First control word	WORD	5
I	First communications information word	WORD	2

C: First Control Word



I: First communications information word

Clear D and I words before executing the instruction.



Note 1 Set the source network address in the routing tables unless you are specifying a local network. Set the source network address to 00 for a local network. If there is more than one CPU Bus Unit mounted in the PLC, the “local network” is the network of the CPU Bus Unit with the lowest unit number.

2 The unit addresses are as follows:

- CPU Unit: 00 hex
- CPU Bus Unit: 10 hex + unit number (hex)
- Special I/O Unit: 20 hex + unit number (hex) (Except for C200H Special I/O Units)
- Inner Board: E1 hex (CS Series only)
- Computer: 01 hex
- Unit connected to network: FE hex (Eliminates the need to specify the unit number of the remote Unit.)

The serial port unit address are as follows:

- Serial Communications Units:
  Port 1: 80 hex + 04 hex × unit number
  Port 2: 81 hex + 04 hex × unit number
- Serial Communications Boards:
  Port 1: E4 hex (228 decimal)
  Port 2: E5 hex (229 decimal)
- CPU Unit
  Peripheral port: FD hex (253 decimal)
  RS-232C port: FC hex (252 decimal)

3 Broadcasting is not possible for RECV2(492). Set 00 hex to request a reception from the local node.

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D																		
C												---						
I																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the serial port number specified in C+1 is not between 0 and 4 hex. • ON if there is no communications port available (i.e., if A211 = 0). • OFF in all other cases.

Function

When RECV2(492) is executed, the number of receive words specified in C starting with the first source word address specified in S at the source node in the source network specified in C are received at the destination node (i.e., the local node). The data is written starting from the word address specified in D1 at the local node.

When the transmission starts, the Execution Completed Flag in I is turned OFF. When a response is returned, the transmission results is shown in the Error Flag in I. The communications response is written to I+1.

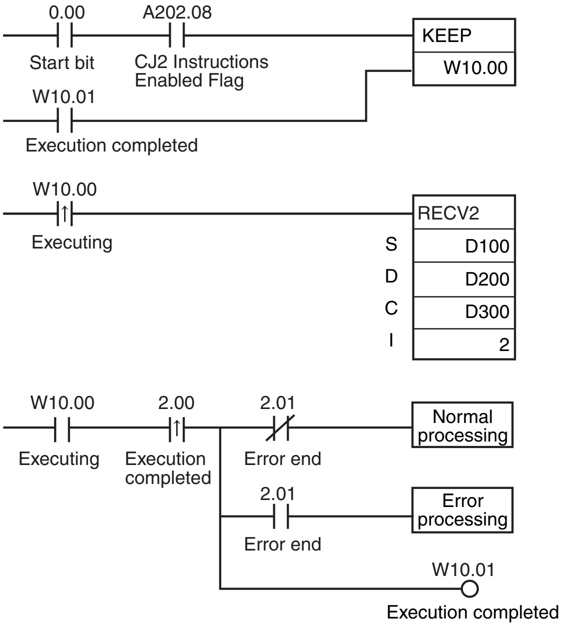
The Communications Port Enabled Flags (A202.00 to A202.07) and Communications Port Error Flags (A219.00 to A219.07) used for the RECV(098) instruction are not used for RECV2(492), which uses the operand I. The communications port completion code (previously stored in A203 to A210) is stored in I+1.

Example Programming

If CIO 0.00 (start bit) and A202.08 (CJ2 Instructions Enabled Flag) are ON, W10.00 (executing) will turn ON.

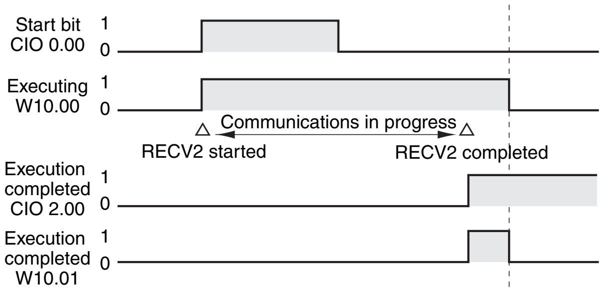
When W10.00 (Executing) turns ON, the ten words of data starting from D100 (D100 to D109) are received from the CPU Unit with node number 3 in the same network and written to ten words starting from D200 (D200 to D209) at the local node. If a response is not received within 10 seconds, the data will be resent up to 3 times.

If communications end normally, CIO 2.00 (execution completed) turns ON, and normal processing is executed. If communications end in an error, CIO 2.01 (error end) turns ON, and error processing is executed.



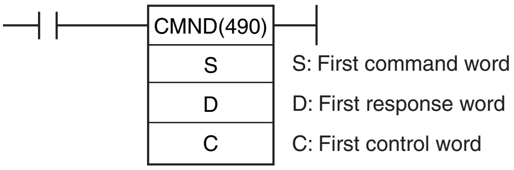
C: D300	0	0	1	0	Number or words to send: 10 words
C+1: D301	0	0	0	0	Receive from the local network and the device itself
C+2: D302	0	3	0	0	Node number 03, unit address 00 (CPU Unit)
C+3: D303	0	0	0	3	Response requested, 3 retries
C+4: D304	0	1	0	0	Response monitoring time: 0064 hexadecimal (10 seconds

● Timing Chart



CMND

Instruction	Mnemonic	Variations	Function code	Function
DELIVER COMMAND	CMND	@CMND	490	Sends an FINS command and receives the response.

	CMND
Symbol	 S: First command word D: First response word C: First control word

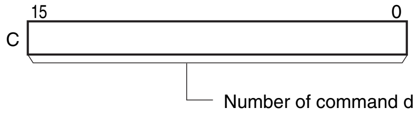
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

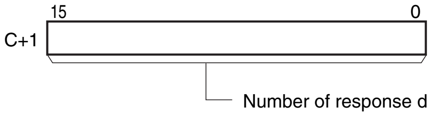
Operands

Operand	Description	Data type	Size
S	First command word	UINT	Variable
D	First response word	UINT	Variable
C	First control word	WORD	6

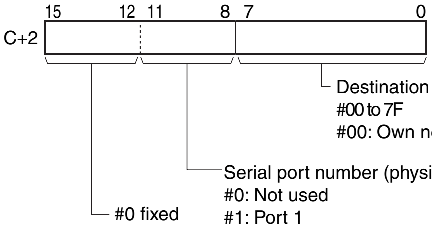
C: First Control Word



Number of command data bytes (See note 1)
0002 to maximum data length (hex)
Number of command data bytes
to own CPU unit is 1990 bytes (#07C6 hex).
For the network, varies by network type.
Example: In the case of Controller Link,
&2 to 1990 (#0002 to 07C6 hex)



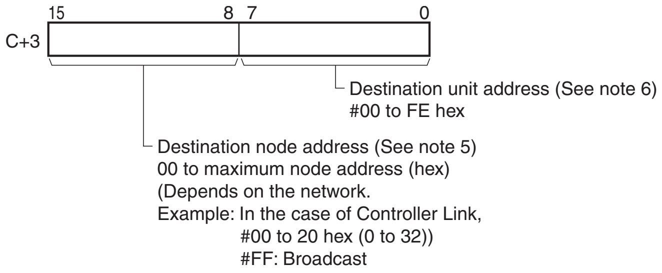
Number of response data bytes (See note 1)
0002 to maximum data length (hex)
Number of response data bytes
to own CPU unit is 1990 bytes (#07C6 hex).
For the network, varies by network type.
Example: In the case of Controller Link,
&0 to 1990 (#0000 to 07C6 hex)

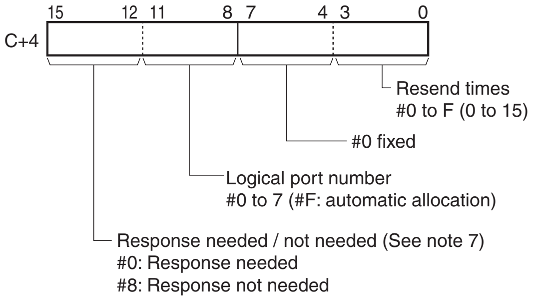


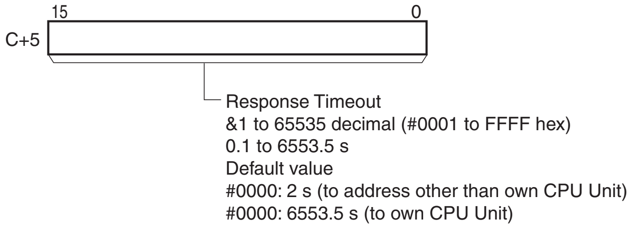
Destination network address (See note 3)
#00 to 7F
#00: Own network (See note 4)

Serial port number (physical port) (See note 2)
#0: Not used
#1: Port 1
#2: Port 2
#3: Reserved
#4: Reserved

#0 fixed







Note 1 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.

If response data longer than the number response data bytes is returned, the response data is not stored. If response data shorter than the number of response data bytes is returned, the response data is received and the remaining area remains unchanged.

2 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).

1) Set the destination unit address (bits 00 to 07 of C+3) to the unit address of the CPU Unit or Serial Communications Unit/Board 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+2, bits 00 to 07)	Unit	Serial port number (C+1, bits 08 to 11)	Serial port
00 hex	CPU Unit	1 hex	Built-in RS-232C port
		2 hex	Peripheral port
10 hex + unit number	Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1
		2 hex	Port 2
E1 hex	Serial Communications Board (Inner Board) (CS Series only)	1 hex	Port 1
		2 hex	Port 2

2) 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.

- 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)

- Serial Communication Board Ports

Port	Port’s unit address
Port 1	E4 hex (228 decimal)
Port 2	E5 hex (229 decimal)

- CPU Unit Ports

Port	Port’s unit address
Peripheral	FD hex (253 decimal)
RS-232C	FC hex (252 decimal)

3 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.

4 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 For a broadcast transmission, set #FF hex.
Set the destination network address to 00 to transmit within the local network.
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).

6 Unit address

- CPU Unit: 00 hex
- CPU Bus Unit: 10 hex + unit number
- Special I/O Unit (except C200H-series Special I/O Units): 20 hex + unit number
- Inner Board: E1 hex (CS Series only)
- Computer: 01 hex
- Unit connected to network: FE hex (not necessary to specify Unit)

Serial port's unit address

- Serial Communications Unit ports
  Port 1: 80 hex + 4 × unit number
  Port 2: 81 hex + 4 × unit number
- Serial Communications Board ports
  Port 1: E4 hex (228 decimal)
  Port 2: E5 hex (229 decimal)
- CPU Unit ports
  Peripheral port: FD hex (253 decimal)
  RS-232C port: FC hex (252 decimal)

7 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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		
C																		

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. • ON if a FINS command is sent to the local CPU Unit while the File Memory Operation Flag (A34313) is ON. • ON if CMND(490) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

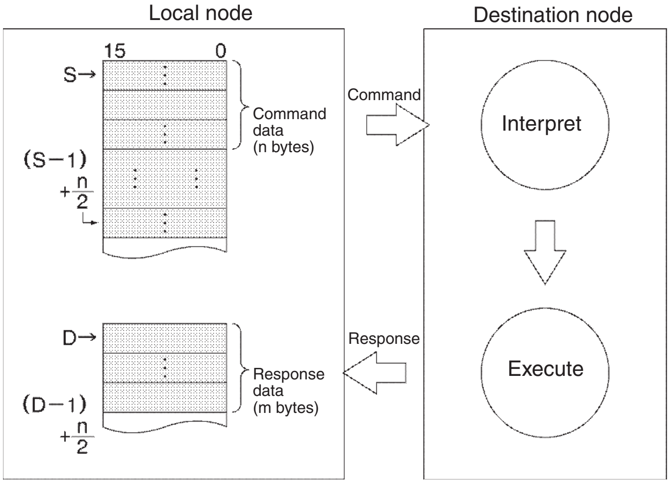
Related Auxiliary Area Words and Bits

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.
File Memory Operation Flag	A343.13	ON when any FINS command is sent to the local CPU Unit (even for FINS commands not related to file memory) or when any of the following instructions or operations are performed for file memory. • FREAD(700) or FWRIT(701) • Program overwrite with control bit in memory • Simple backup operation

Note Refer to the FINS command response codes in the *CS/CJ Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Function

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.

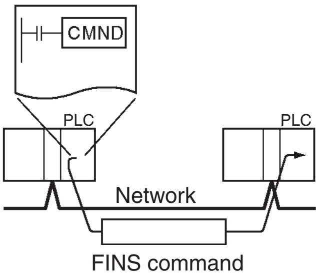


- If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when CMND(490) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A202.00 to A202.07) and Communications Port Error Flag (ports 00 to 07: A219.00 to A219.07) will be turned OFF and 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.
- 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.)
- CMND(490) can be used to transmit command data to a particular serial port in the destination device as well as the device itself.
- The CPU Unit executing CMND(490) can send a FINS command to itself (except for CS-series CS1 CPU Units prior to V1□). Use the following control data settings to achieve this.
  - Destination network address (bits 00 to 07 of C+2): 00 hex (local network)
  - Serial port No. (bits 08 to 11 of C+2): 0 hex (not used)
  - Destination unit address (bits 00 to 07 of C+3): 00 hex (CPU Unit)

  - Destination node address (bits 08 to 15 of C+3): 00 hex (local node)
  - Number of retries (bits 00 to 03 of C+4): 0 hex (this setting is invalid; set it to 0)
  - Response monitoring time: (bits 00 to 15 of C+5): 0000 to FFFF hex (but 0000 will specify 6553.5 s, and not 2 s as normal)
- 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.”
- When broadcast is specified, there is no response and no resending.
- 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 (when set to host link mode) as well as a PLC (CPU Unit, Inner Board (CS Series only), or CPU Bus Unit) or computer connected through a Controller Link or Ethernet network.

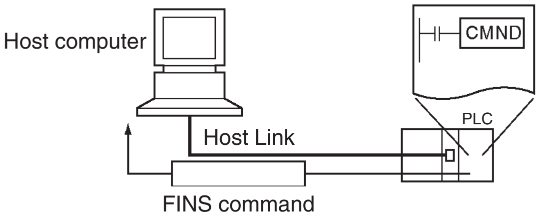
● Transmission through the Network

CMND(490) can be used to transmit any FINS command to a personal computer or a PLC (CPU Unit, Inner Board (CS Series only), or CPU Bus Unit) connected by a Controller Link network or Ethernet link.



● Transmission through Host Link

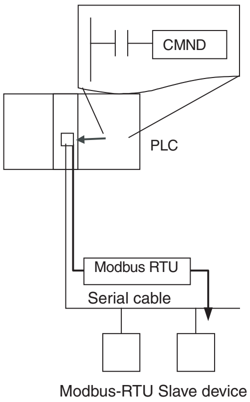
When the CPU Unit's built-in serial port, a Serial Communications Board (CS Series only), or Serial Communications Unit is in host link mode and connected one-to-one with a host computer, CMND(490) can be executed to transmit any FINS command from the PLC to the host computer the next time that the PLC has the right to transmit. It is also possible to transmit to other host computers connected to other PLCs elsewhere in the network.



● Issuing a FINS command to a device (a component or a PLC of a Host Link slave) connected to a serial port (Serial Gateway)

- 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).

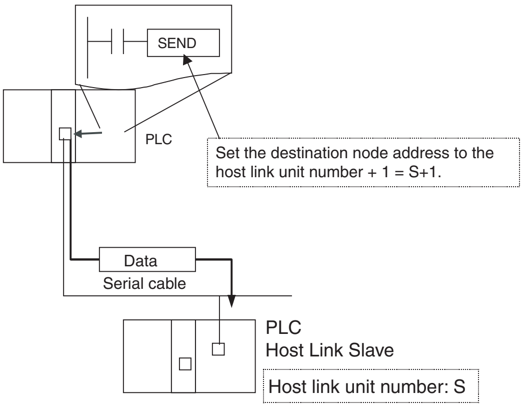


Convert to CompoWay/F command: 2803 hex
Convert to Modbus-RTU command: 2804 hex
Convert to Modbus-ASCII command: 2805 hex

Note The Modbus-RTU and Modbus-ASCII commands cannot be sent to the CPU Unit’s serial ports.

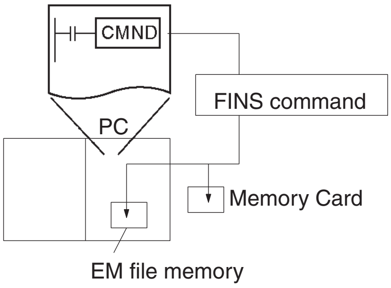
- Sending to a PLC operating as a Host Link Slave

  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 Communications Board or Unit's serial port. In this case, the destination node address must be set to the host link unit number + 1.



● Sending a FINS Command to the CPU Unit Executing CMND(490)

The CPU Unit executing CMND(490) can send a FINS command to itself (excluding CS-series CS1 CPU Units without a suffix of -V☐). For example, file memory commands (command codes 22☐☐ hex) can be sent to format file memory, delete files, copy files, and perform other operations. Refer to *5-2 Manipulating Files* of the *CS/CJ-series CPU Unit Programming Manual* or *7-3 File Memory Operations* of the *CJ2 CPU Unit Operation Manual* for details.



The File Memory Operation Flag (A343.13) will turn ON when any FINS command is sent to the local CPU Unit (even for FINS commands not related to file memory). Always use A343.13 in an NC input condition for CMND(490) to ensure that only one FINS command is being executed for the CPU Unit at the same time.

Hint

- 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).
- Issuing a FINS command (non-solicited communications) to a host computer connected by a host link

  CMND(490) can be executed for the either port on the CPU Unit, a Serial Communications Board (CS Series only), or 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 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 (CPU Unit), E1 (Inner Board, CS Series only), or unit number + 10 hexadecimal (Serial Port Unit).

Precaution

- 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.

Note Refer to *Automatically Allocating Communications Ports (i.e., Internal Logical Ports)* in *SECTION 3 Instructions - Network Instructions* for details on using automatic allocation of the communications port number (logical port).

- 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.
- Always use one of the Communications Port Enabled Flags (A202.00 to A202.07) in an NO input condition and the File Memory Operation Flag (A343.13) in an NC input condition for CMND(490) when send a FINS command to the local CPU Unit.
- 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.

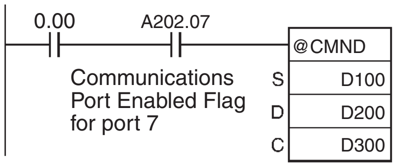
Example Programming

● Issuing a FINS Command to Another CPU Unit on the Network

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 14 words or 28 bytes.



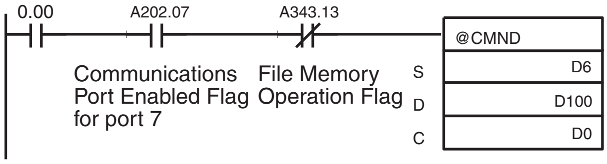
	15 8	7 0	
S: D100	0 1	0 1	Command code: 0101 hexadecimal (MEMORY AREA READ)
S+1:D101	8 2	0 0	D10 (Data area = 82 hexadecimal, address = 000A00)
S+2:D102	0 A	0 0	
S+3:D103	0 0	0 A	Number of words to read = 0A hexadecimal (10 decimal)

	15 8	7 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)

● Issuing a FINS Command to Own CPU Unit

The following program section shows an example of sending a FINS command to the local CPU Unit.

When CIO 0.00 and A202.07 (the Communications Port Enabled Flag for port 07) are ON and A343.13 (File Memory Operation Flag) is OFF, CMND(490) transmits FINS command 2215 (CREATE/DELETE DIRECTORY) to the local CPU Unit. The response is stored in D100 to D101. Here, the FINS command will create a directory called CS/CJ under the OMRON directory. The command code (2 bytes) and the end code (2 bytes) will be returned and stored as the response.

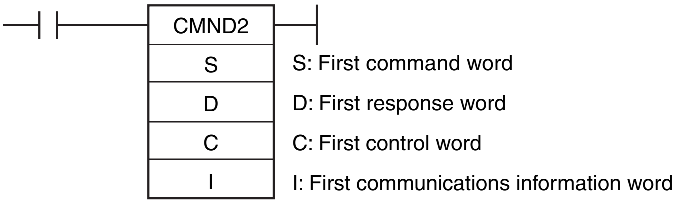


		15 8	7 0	
S:	D6	2 2	1 5	Command code: 2215 hex (CREATE/DELETE DIRECTORY)
S+1:	D7	8 0	0 0	Disk No.: 8000 hex (Memory Card)
S+2:	D8	0 0	0 0	Parameter: 0000 hex (create directory)
S+3:	D9	4 3	5 3	Subdirectory name: CS1□□□□□.□□□ (□= space)
S+4:	D10	3 1	2 0	
S+5:	D11	2 0	2 0	
S+6:	D12	2 0	2 0	
S+7:	D13	2 E	2 0	
S+8:	D14	2 0	2 0	
S+9:	D15	0 0	0 6	Directory name length: 0006 (6 characters)
S+10:	D16	5 C	4 F	Absolute directory path: \OMRON
S+11:	D17	4 D	5 2	
S+12:	D18	4 F	4 E	

		15 8	7 0	
S:	D0	0 0	1 A	Bytes of command data: 001A (26 decimal)
S+1:	D1	0 0	0 4	Bytes of response data: 0004 (4)
S+2:	D2	0 0	0 0	Destination network address: 00 hex (local network)
S+3:	D3	0 0	0 0	Destination unit address: 00 hex, Destination node number: 00 hex (CPU Unit at local node)
S+4:	D4	0 7	0 0	Response requested, port number 7, 0 retries
S+5:	D5	0 0	0 0	Response monitoring time = 0000 hex (6,553.5 seconds)

CMND2

Instruction	Mnemonic	Variations	Function code	Function
DELIVER COMMAND 2	CMND2	@CMND2	493	Sends an FINS command and receives the response.

	CMND2
Symbol	 S: First command word D: First response word C: First control word I: First communications information word

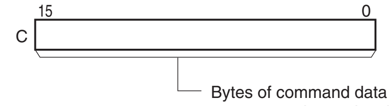
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

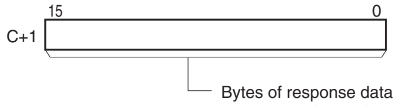
Operands

Operand	Description	Data type	Size
S	First command word	WORD	Variable
D	First response word	WORD	Variable
C	First control word	WORD	6
I	First communications information word	WORD	2

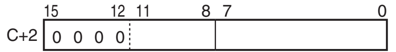
C: First Control Word



Bytes of command data (See note 1.)
0002 to maximum data length (hex)
The maximum data length for command addressed to the local CPU Unit is 1,990 bytes (07C6 hex).
For commands sent via networks, the maximum data length depends on the network, for example, 2 to 1,990 decimal (0002 to 07C6 hex) for Controller Link.



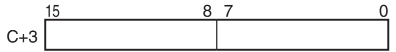
Bytes of response data (See note 1.)
0000 to maximum data length (hex)
The maximum data length for responses received from the local CPU Unit is 1,990 bytes (07C6 hex).
For responses received via networks, the maximum data length depends on the network, for example, 0 to 1,990 decimal (0000 to 07C6 hex) for Controller Link.



Destination network address
#00 to #7F hex (0 to 127)
#00: Local network

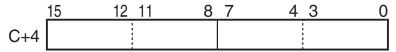
Serial port number (physical port)
#0: Do not use.
#1: Port 1
#2: Port 2
#3: Reserved
#4: Reserved

Always #0.



Destination unit address (See note 2.)
00 to FE hex

Destination node number (See note 3.)
00 to maximum node number (hex)
(Depends on network, for example
00 to 20 hex (0 to 32 decimal) for Controller Link.)
FF hex: Broadcast

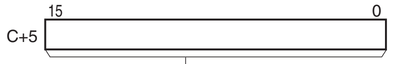


Number of retries
0 to F hex (0 to 15 decimal)

Always 0.

Communications port number
0 to 7 hex (F hex: automatic port allocation)

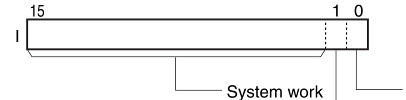
Response setting
0: Response requested
8: No response requested



Response monitoring time
1 to 65535 decimal (0001 to FFFF hex)
0.1 to 6,553.5 s
Default value
0000 hex: 2 s (to address other than own CPU Unit)
0000 hex: 6553.5 s (to own CPU Unit)

I: First communications information word

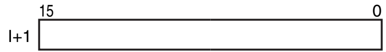
Clear D and I words before executing the instruction.



System work area

Execution Completed Flag
0: Not completed.
1: Execution completed

Error Flag
0: No error
1: Error



Communications completion code: 0000 to FFFF hex

Note 1 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. If more bytes are received than the specified number of bytes, only the specified number of bytes will be stored. If fewer bytes are received, they will be stored and the remaining words in the reception area will not be changed.

2 The unit addresses are as follows:

- CPU Bus Unit: 10 hex + unit number
- Special I/O Unit (except for C200H Special I/O Units): 20 hex + unit number
- Inner Board: E1 hex (CS Series only)
- Computer: 01 hex
- Unit connected to network: FE hex (Eliminates the need to specify the unit number of the remote Unit.)

The serial port unit address are as follows:

- Serial Communications Units:
  Port 1: 80 hex + 04 hex × unit number
  Port 2: 81 hex + 04 hex × unit number
- Serial Communications Boards:
  Port 1: E4 hex (228 decimal)
  Port 2: E5 hex (229 decimal)
- CPU Unit
  Peripheral port: FD hex (253 decimal)
  RS-232C port: FC hex (252 decimal)

3 Set FF hex for broadcasting. Set 00 hex to sent to the local node. When specifying a serial port using a serial gateway (converting to Host Link FINS), set the destination node address to one higher than the Host Link unit number of the destination PLC (1 to 32).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D																		
C												---						
I																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the serial port number specified in C+1 is not between 0 and 4 hex. • ON if there is no communications port available (i.e., if A211 = 0). • OFF in all other cases.

Function

CMND2(493) transfers the specified number of bytes of FINS send command data beginning at word address S to the device with the designated unit address through the PLC's CPU Bus or over a network. The response is stored in memory beginning at word D1.

CMND2(493) can be used to send an FINS command to the local CPU Unit as well.

This is mainly done to send FINS commands for file memory operations.

The File Memory Operation Flag (A343.13) will turn ON when any FINS command is sent to the local CPU Unit (even for FINS commands not related to file memory). Always use A343.13 in an NC input condition for CMND2(493) to ensure that only one FINS command is being executed for the CPU Unit at the same time.

This means that command cannot be simultaneously sent to the local CPU Unit using both the CMND(490) and CMND2(493) instructions.

When sending starts, the Execution Completed Flag in I is turned OFF. When a response is returned, the results of the send is shown in the Error Flag in I. The communications response is written to I+1.

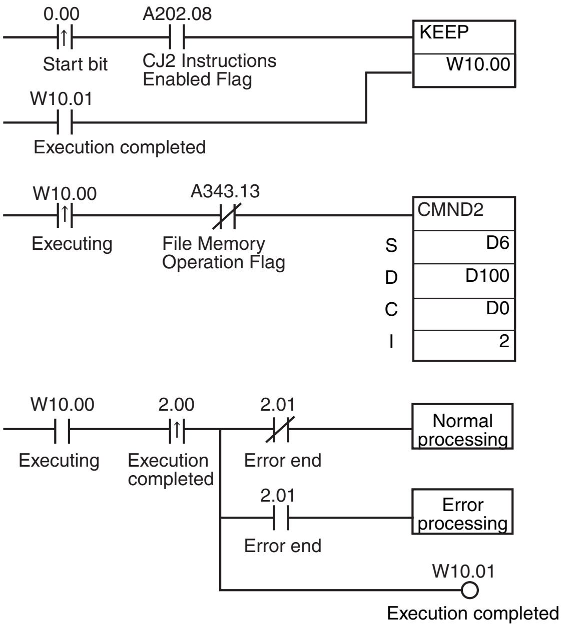
The Communications Port Enabled Flags (A202.00 to A202.07) and Communications Port Error Flags (A219.00 to A219.07) used for the CMND(492) instruction are not used for CMND2(493), which uses the operand I. The communications port completion code (previously stored in A203 to A210) is stored in I+1.

Example Programming

If CIO 0.00 (start bit) and A202.08 (CJ2 Instructions Enabled Flag) are ON, W10.00 (executing) will turn ON. If W10.00 (executing) is ON and A343.13 (Accessing File Data Flag) is OFF, FINS command 2215 hex (CREATE/DELETE SUBDIRECTORY) is set to the local CPU Unit and the response is stored in D100 and D101.

The FINS command creates a director called “CS1” in the “OMRON” directory in the Memory Card in the CPU Unit. The command code 2215 hex (2 bytes) and the 2-byte end code are returned as the response.

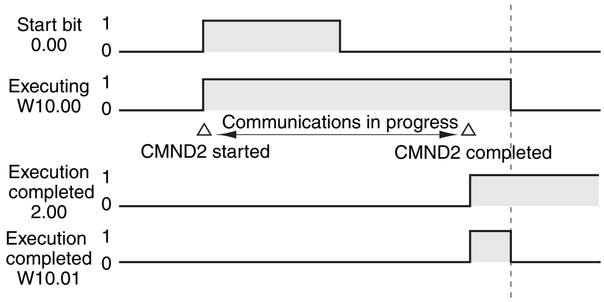
If communications end normally, CIO 2.00 (execution completed) turns ON, and normal processing is executed. If communications end in an error, CIO 2.01 (error end) turns ON, and error processing is executed.



Operand	Word	15–8	7–0	Contents
S:	D6	2 2	1 5	Command code = 2215 hex (CREATE/DELETE SUBDIRECTORY)
S+1:	D7	8 0	0 0	Disk No. = 8000 hex (Memory Card)
S+2:	D8	0 0	0 0	Parameter= 0000 hex (create directory)
S+3:	D9	4 3	5 3	Subdirectory name = 'CS1□□□□□. □□□' (□: Space code)
S+4:	D10	3 1	2 0	
S+5:	D11	2 0	2 0	
S+6:	D12	2 0	2 0	
S+7:	D13	2 E	2 0	
S+8:	D14	2 0	2 0	
S+9:	D15	0 0	0 6	Directory length = 0006 hex (6 characters)
S+10:	D16	5 C	4 F	Absolute directory path = '\OMRON'
S+11:	D17	4 D	5 2	
S+12:	D18	4 F	4 E	

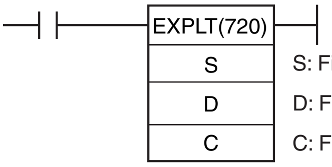
Operand	Word	15–8	7–0	Contents
C:	D0	0 0	1 A	Bytes of command data = 001A hex (26 bytes)
C+1:	D1	0 0	0 4	Bytes of response data = 0004 hex (4 bytes)
C+2:	D2	0 0	0 0	Destination address = 0000 hex (local network)
C+3:	D3	0 0	0 0	00 hex (local node), 00 hex (CPU Unit)
C+4:	D4	0 0	0 0	Response required, 0 retries
C+5:	D5	0 0	0 0	Response monitoring time = 0000 hex (6,553.5 s)

● Timing Chart



EXPLT

Instruction	Mnemonic	Variations	Function code	Function
EXPLICIT MESSAGE SEND	EXPLT	@EXPLT	720	Sends an explicit message with any service code.

	EXPLT
Symbol	 S: First word of send message D: First word of received message C: First control word

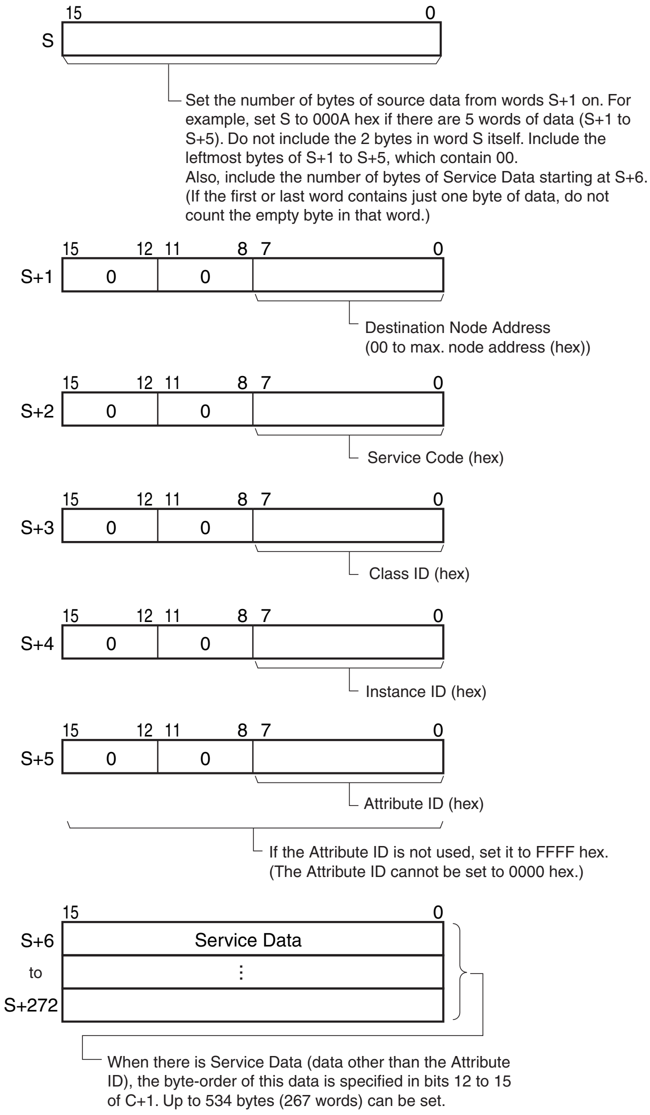
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First word of send message	WORD	Variable
D	First word of received message	WORD	Variable
C	First control word	LWORD	4

S: First word of send message



Set the number of bytes of source data from words S+1 on. For example, set S to 000A hex if there are 5 words of data (S+1 to S+5). Do not include the 2 bytes in word S itself. Include the leftmost bytes of S+1 to S+5, which contain 00.
Also, include the number of bytes of Service Data starting at S+6.
(If the first or last word contains just one byte of data, do not count the empty byte in that word.)

Destination Node Address
(00 to max. node address (hex))

Service Code (hex)

Class ID (hex)

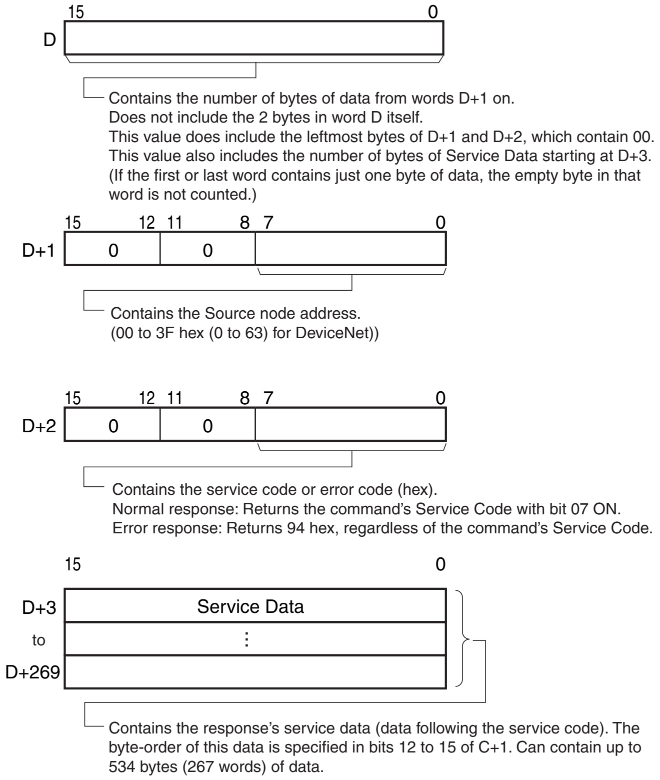
Instance ID (hex)

Attribute ID (hex)

If the Attribute ID is not used, set it to FFFF hex.
(The Attribute ID cannot be set to 0000 hex.)

When there is Service Data (data other than the Attribute ID), the byte-order of this data is specified in bits 12 to 15 of C+1. Up to 534 bytes (267 words) can be set.

D: First word of received message



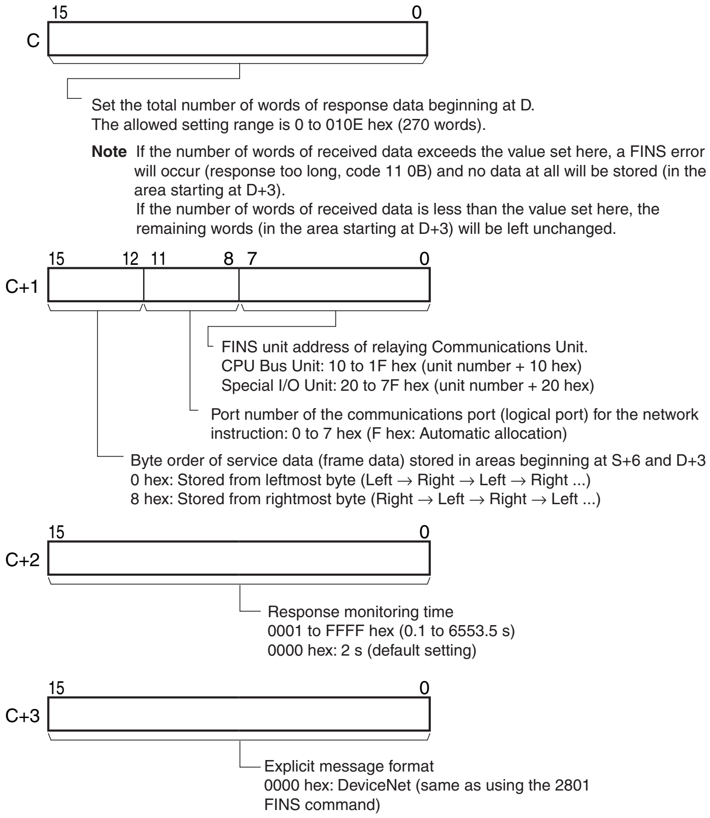
Contains the number of bytes of data from words D+1 on.
Does not include the 2 bytes in word D itself.
This value does include the leftmost bytes of D+1 and D+2, which contain 00.
This value also includes the number of bytes of Service Data starting at D+3.
(If the first or last word contains just one byte of data, the empty byte in that word is not counted.)

Contains the Source node address.
(00 to 3F hex (0 to 63) for DeviceNet))

Contains the service code or error code (hex).
Normal response: Returns the command's Service Code with bit 07 ON.
Error response: Returns 94 hex, regardless of the command's Service Code.

Contains the response's service data (data following the service code). The byte-order of this data is specified in bits 12 to 15 of C+1. Can contain up to 534 bytes (267 words) of data.

C: First control word



Set the total number of words of response data beginning at D.
The allowed setting range is 0 to 010E hex (270 words).

Note If the number of words of received data exceeds the value set here, a FINS error will occur (response too long, code 11 0B) and no data at all will be stored (in the area starting at D+3).
If the number of words of received data is less than the value set here, the remaining words (in the area starting at D+3) will be left unchanged.

FINS unit address of relaying Communications Unit.
CPU Bus Unit: 10 to 1F hex (unit number + 10 hex)
Special I/O Unit: 20 to 7F hex (unit number + 20 hex)

Port number of the communications port (logical port) for the network instruction: 0 to 7 hex (F hex: Automatic allocation)

Byte order of service data (frame data) stored in areas beginning at S+6 and D+3
0 hex: Stored from leftmost byte (Left → Right → Left → Right ...)
8 hex: Stored from rightmost byte (Right → Left → Right → Left ...)

Response monitoring time
0001 to FFFF hex (0.1 to 6553.5 s)
0000 hex: 2 s (default setting)

Explicit message format
0000 hex: DeviceNet (same as using the 2801 FINS command)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		
C																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. • OFF in all other cases.

Related Auxiliary Area Words and Bits

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 *About Explicit Message Instructions*.

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.

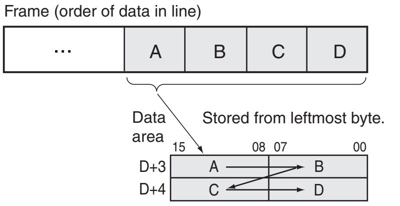
Function

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.

Note 1 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.)

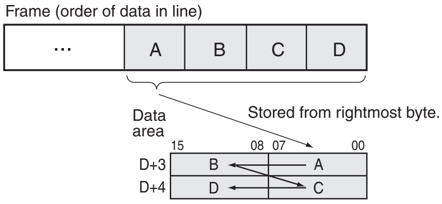
2 The number of bytes of received data in D includes the 4 bytes in D+1 and D+2 as well as the number of bytes of service data beginning at D+3. (For example, if there is 1 byte of service data, there are 5 bytes of data all together and D contains 0005 hex.)
The setting in bits 12 to 15 of C+1 (0 or 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.

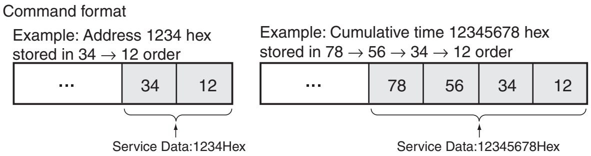
- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Precaution

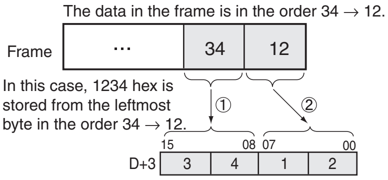
Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.



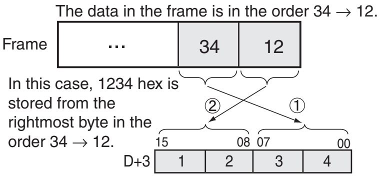
The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

1) **Data in 2-byte Units**

- Storing Data from the Leftmost Byte (Bits 12 to 15 of C+1 = 0 hex)
  Example: Storing the value 1234 hex in D+3

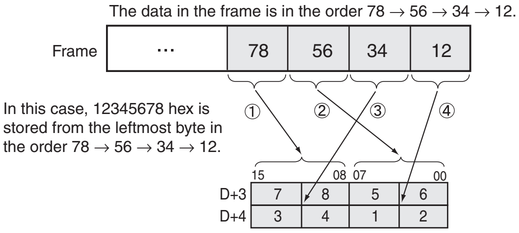


- Storing Data from the Rightmost Byte (Bits 12 to 15 of C+1 = 8 hex)
  Example: Storing the value 1234 hex in D+3

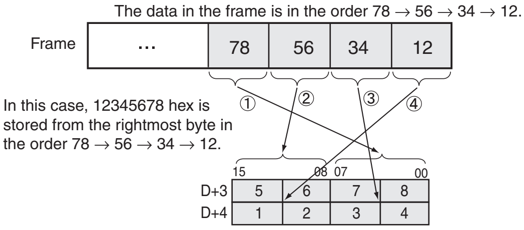


2) **Data in 4-byte Units**

- Storing Data from the Leftmost Byte (Bits 12 to 15 of C+1 = 0 hex)
  Example: Storing the value 12345678 hex in D+3 and D+4



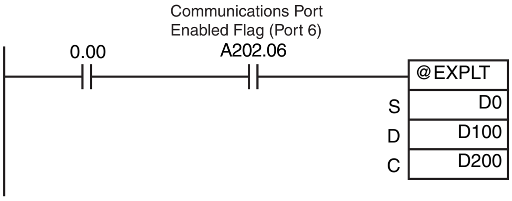
- Storing Data from the Rightmost Byte (Bits 12 to 15 of C+1 = 8 hex)
  Example: Storing the value 12345678 hex in D+3 and D+4



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 Programming

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).

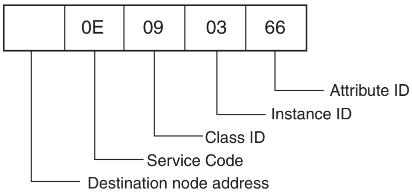


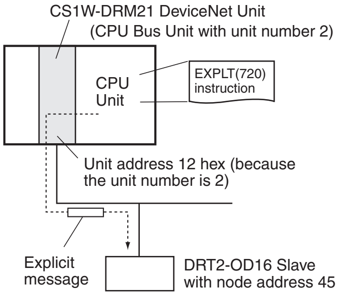
When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, EXPLT(720) reads the Total ON Time (s) or Number of Contact Operations from a DRT2 Slave (I/O Terminal). In this case, the Total ON Time or Number of Contact Operations for input 3 are read.

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.

Explicit message command format





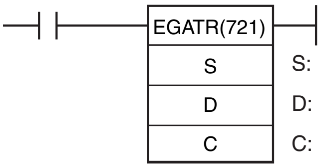
S:	D0	0	0	0	A	Number of bytes of data: S+1 to S+5 = 5 words = 10 bytes = 0A hex
S+1:	D1	0	0	2	D	Slave's node address = 45 = 2D hex
S+2:	D2	0	0	0	E	Service Code = 0E hex
S+3:	D3	0	0	0	9	Class ID = 09 hex
S+4:	D4	0	0	0	3	Instance ID = 03 hex (Input 3)
S+5:	D5	0	0	6	6	Attribute ID = 66 hex

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)

EGATR

Instruction	Mnemonic	Variations	Function code	Function
EXPLICIT GET ATTRIBUTE	EGATR	@EGATR	721	Sends an information/status read command in an explicit message (Get Attribute Single, Service Code: 0E hex).

	EGATR
Symbol	 S: First word of send message D: First word of received message C: First control word

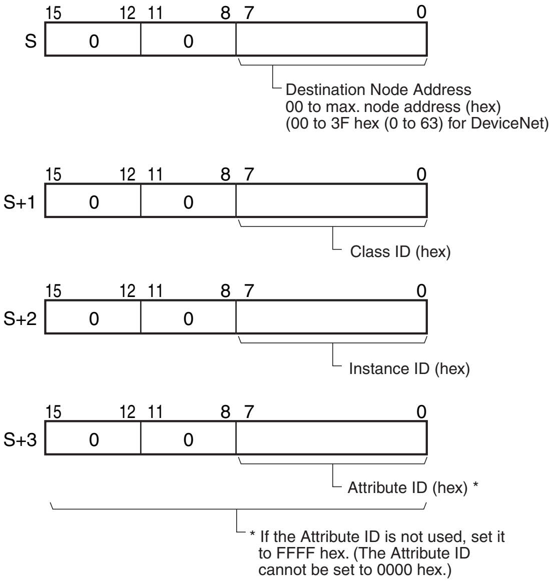
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First word of send message	ULINT	4
D	First word of received message	WORD	Variable
C	First control word	LWORD	4

S: First word of send message



Destination Node Address
00 to max. node address (hex)
(00 to 3F hex (0 to 63) for DeviceNet)

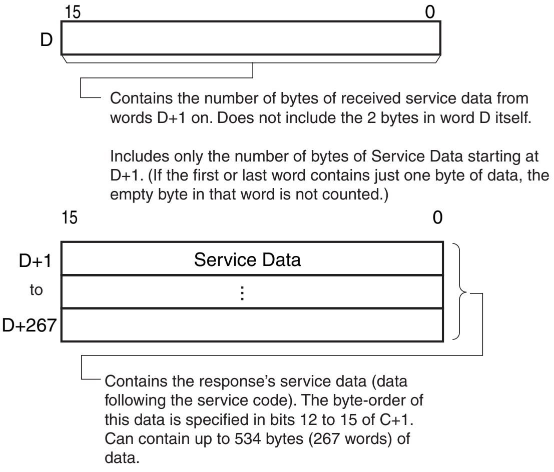
Class ID (hex)

Instance ID (hex)

Attribute ID (hex) *

* If the Attribute ID is not used, set it to FFFF hex. (The Attribute ID cannot be set to 0000 hex.)

D: First word of received message

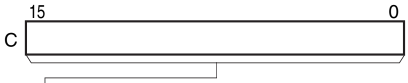


Contains the number of bytes of received service data from words D+1 on. Does not include the 2 bytes in word D itself.

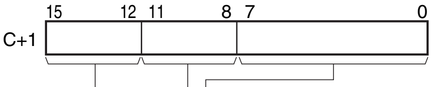
Includes only the number of bytes of Service Data starting at D+1. (If the first or last word contains just one byte of data, the empty byte in that word is not counted.)

Contains the response's service data (data following the service code). The byte-order of this data is specified in bits 12 to 15 of C+1. Can contain up to 534 bytes (267 words) of data.

C: First control word



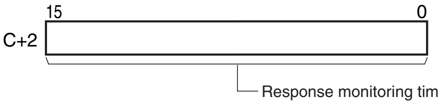
Set the maximum number of words of data in the received data beginning at D.
The allowed setting range is 0 to 010C hex (268 words).
If the number of words of received data exceeds the value set here, a FINS error will occur (response too long, code 11 0B) and no data at all will be stored (in the area starting at D+3).
If the number of words of received data is less than the value set here, the remaining words (in the area starting at D+3) will be left unchanged.



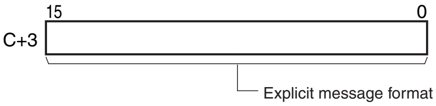
FINS unit address of relaying Communications Unit.
CPU Bus Unit: 10 to 1F hex (unit number + 10 hex)
Special I/O Unit: 20 to 7F hex (unit number + 20 hex)

Port number of the communications port (logical port) for the network instruction: 0 to 7 hex (F hex: Automatic allocation)

Byte order of service data (frame data) stored in areas beginning at S+6 and D+3
0 hex: Stored from leftmost byte (Left → Right → Left → Right ...)
8 hex: Stored from rightmost byte (Right → Left → Right → Left ...)



Response monitoring time
0001 to FFFF hex (0.1 to 6553.5 s)
0000 hex: 2 s (default setting)



Explicit message format
0000 hex: DeviceNet (same as using the 2801 FINS command)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		
C																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. • OFF in all other cases.

Related Auxiliary Area Words and Bits

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 *About Explicit Message Instructions*.

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.

Function

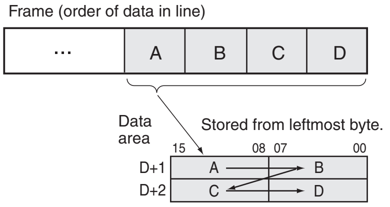
Sends the “read information/status” explicit message command (stored in words S+1 to S+3) to the node address specified in S, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C+1.

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.

Note 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.)

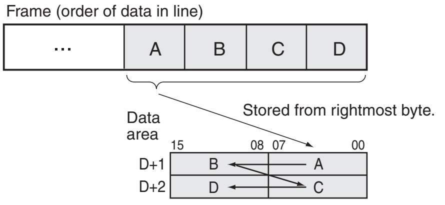
The setting in bits 12 to 15 of C+1 (0 or 8 hex) determines the byte-order of the service data stored at S+6 and D+3.

- 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.

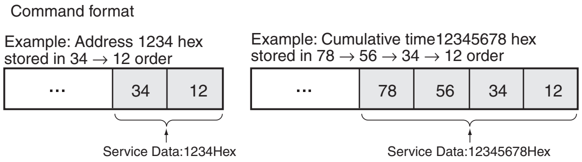
- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Precaution

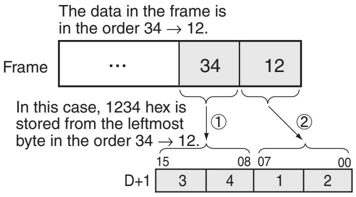
Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.



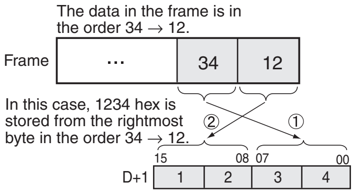
The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

1) Data in 2-byte Units

- Storing Data from the Leftmost Byte (Bits 12 to 15 of C+1 = 0 hex)
  Example: Storing the value 1234 hex in D+1

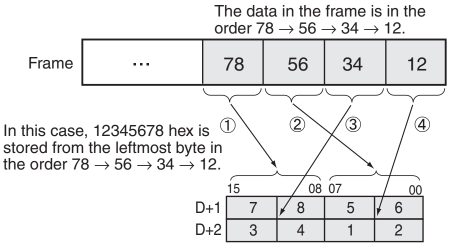


- Storing Data from the Rightmost Byte (Bits 12 to 15 of C+1 = 8 hex)
  Example: Storing the value 1234 hex in D+1

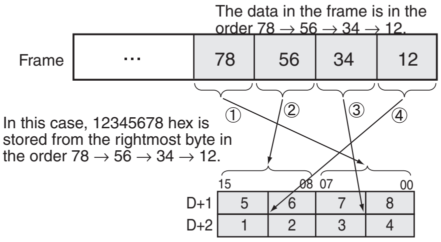


2) Data in 4-byte Units

- Storing Data from the Leftmost Byte (Bits 12 to 15 of C+1 = 0 hex)
  Example: Storing the value 12345678 hex in D+1 and D+2

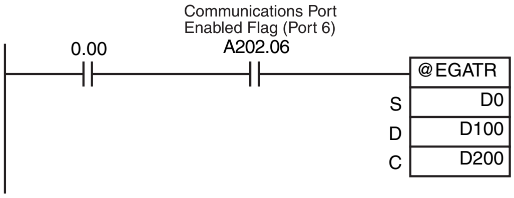


- Storing Data from the Rightmost Byte (Bits 12 to 15 of C+1 = 8 hex)
  Example: Storing the value 12345678 hex in D+1 and D+2



Example Programming

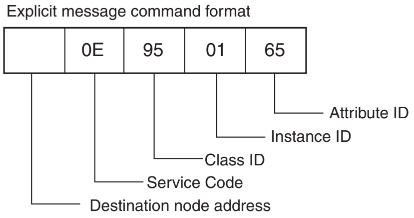
In this example, EGATR(721) is used to read the general status of a DRT2 Slave (I/O Terminal).

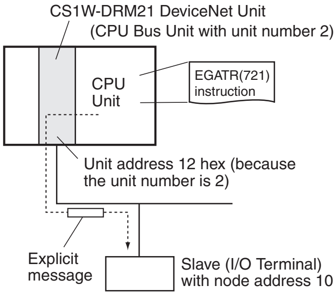


When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, EGATR(721) reads the general status of the DRT2 Slave (I/O Terminal). In this case, the Total ON Time or Number of Contact Operations for input 3 are read.

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.

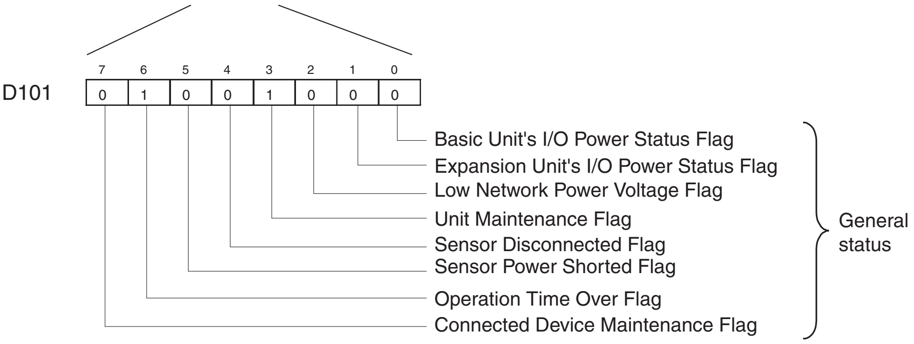




S:	D0	0	0	0	A	Slave's node address = 10 = 0A hex
S+1:	D1	0	0	9	5	Class ID = 95 hex
S+2:	D2	0	0	0	1	Instance ID = 01 hex
S+3:	D3	0	0	6	5	Attribute ID = 65 hex

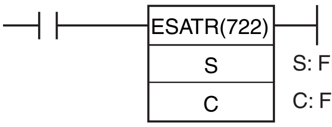
C:	D200	0	0	0	2	Set 2 words = 0002 hex since there are 2 words in D to D+1.
C+1:	D201	8	6	1	2	Byte order = 8 hex (from rightmost byte), communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+2:	D202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+3:	D203	0	0	0	0	Explicit 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	



ESATR

Instruction	Mnemonic	Variations	Function code	Function
EXPLICIT SET ATTRIBUTE	ESATR	@ESATR	722	Sends an information write command in an explicit message (Set Attribute Single, Service Code: 10 hex).

	ESATR
Symbol	 S: First word of send message C: First control word

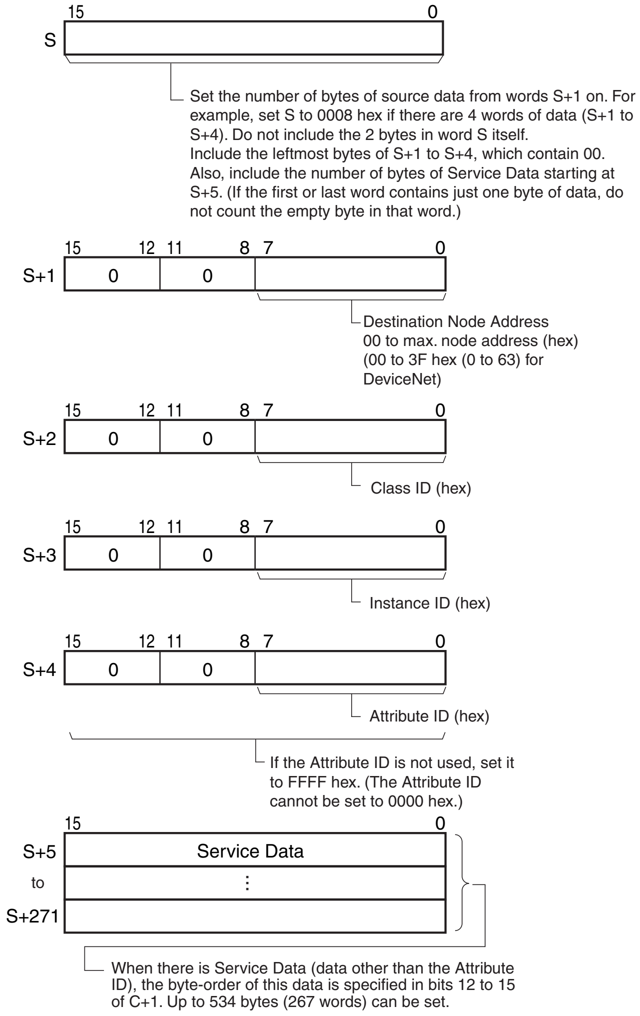
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First word of send message	WORD	Variable
C	First control word	WORD	3

S: First word of send message



Set the number of bytes of source data from words S+1 on. For example, set S to 0008 hex if there are 4 words of data (S+1 to S+4). Do not include the 2 bytes in word S itself. Include the leftmost bytes of S+1 to S+4, which contain 00. Also, include the number of bytes of Service Data starting at S+5. (If the first or last word contains just one byte of data, do not count the empty byte in that word.)

Destination Node Address
00 to max. node address (hex)
(00 to 3F hex (0 to 63) for DeviceNet)

Class ID (hex)

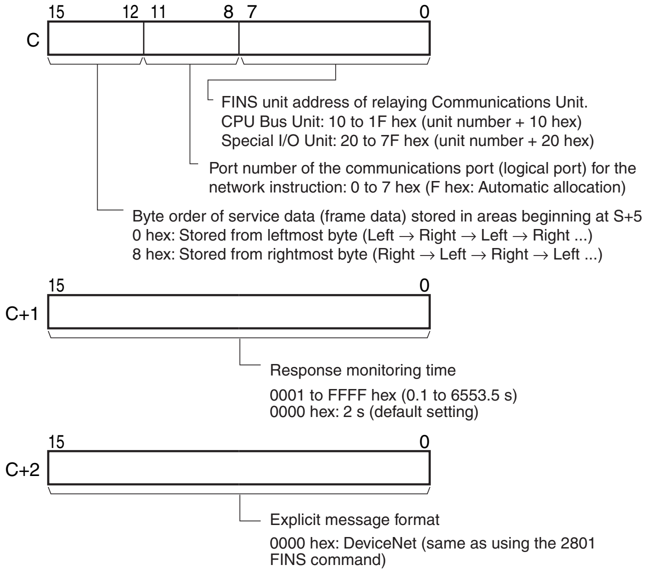
Instance ID (hex)

Attribute ID (hex)

If the Attribute ID is not used, set it to FFFF hex. (The Attribute ID cannot be set to 0000 hex.)

When there is Service Data (data other than the Attribute ID), the byte-order of this data is specified in bits 12 to 15 of C+1. Up to 534 bytes (267 words) can be set.

C: First control word



FINS unit address of relaying Communications Unit.
CPU Bus Unit: 10 to 1F hex (unit number + 10 hex)
Special I/O Unit: 20 to 7F hex (unit number + 20 hex)

Port number of the communications port (logical port) for the network instruction: 0 to 7 hex (F hex: Automatic allocation)

Byte order of service data (frame data) stored in areas beginning at S+5
0 hex: Stored from leftmost byte (Left → Right → Left → Right ...)
8 hex: Stored from rightmost byte (Right → Left → Right → Left ...)

Response monitoring time
0001 to FFFF hex (0.1 to 6553.5 s)
0000 hex: 2 s (default setting)

Explicit message format
0000 hex: DeviceNet (same as using the 2801 FINS command)

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
C																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. • OFF in all other cases.

Related Auxiliary Area Words and Bits

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 *About Explicit Message Instructions*.

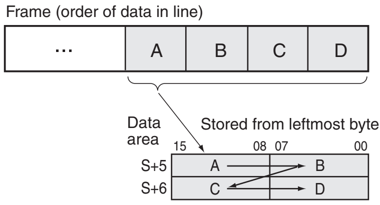
Name	Address	Operation
Communications Port Enabled Flag	A202.00 to A202.07	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding 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.

Function

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.

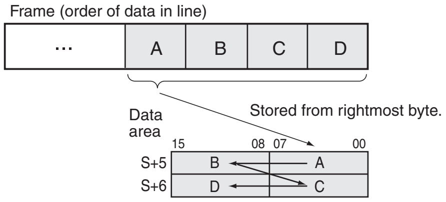
The setting in bits 12 to 15 of C (0 or 8 hex) determines the byte-order of the service data stored at S+5.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C+1 to 0 hex.



Note: A, B, C, and D represent bytes of data.

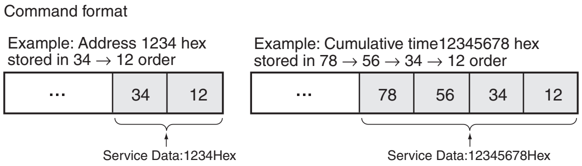
- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Precaution

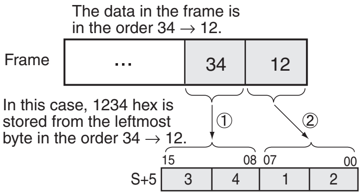
Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.



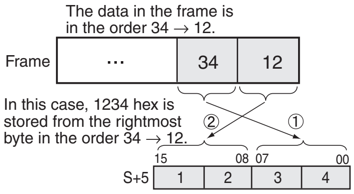
The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

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

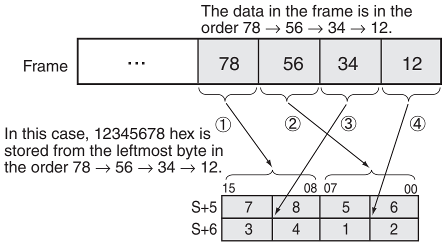


- Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
  Example: Storing the value 1234 hex in S+5

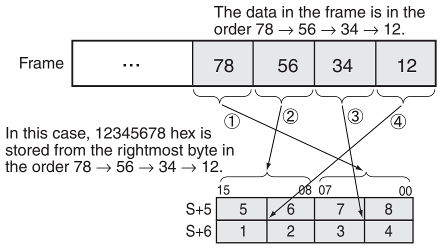


2) Data in 4-byte Units

- 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

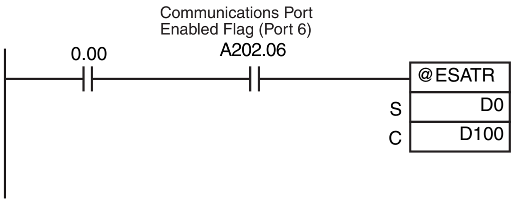


- Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
  Example: Storing the value 12345678 hex in S+5 and S+6



Example Programming

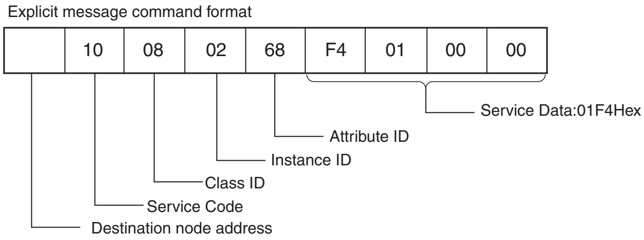
In this example, ESATR(722) is used to overwrite the Number of Contact Operations set value in a DRT2 Slave (I/O Terminal).

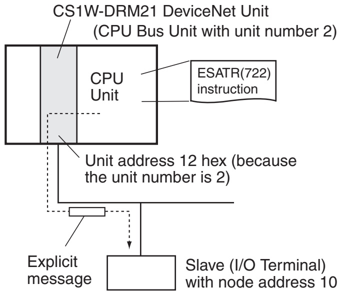


When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, EXPLT(720) writes the Number of Contact Operations set value for input 2 in a DRT2 Slave (I/O Terminal).

(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	

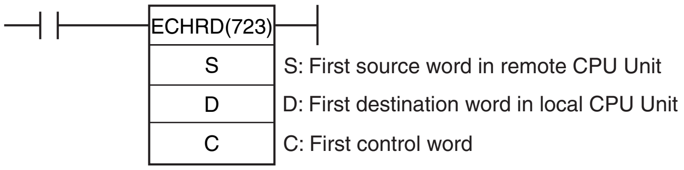
C:	D201	8	6	1	2	Byte order = 8 hex (from rightmost byte), communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+1:	D202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+2:	D203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

ECHRD

Instruction	Mnemonic	Variations	Function code	Function
EXPLICIT WORD READ	ECHRD	@ECHRD	723	Reads data to the local CPU Unit from another CPU Unit in the network. (The remote CPU Unit must support explicit messages.)

Symbol

ECHRD



S: First source word in remote CPU Unit
D: First destination word in local CPU Unit
C: First control word

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word in remote CPU Unit	UINT	1
D	First destination word in local CPU Unit	UINT	1
C	First control word	WORD	5

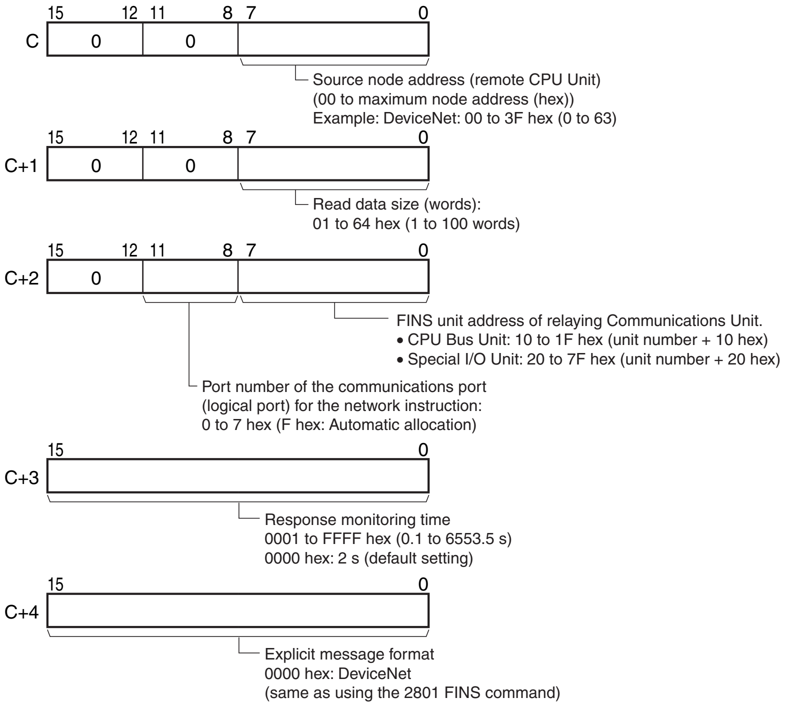
S: First Source Word in Remote CPU Unit

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



Source node address (remote CPU Unit)
(00 to maximum node address (hex))
Example: DeviceNet: 00 to 3F hex (0 to 63)

Read data size (words):
01 to 64 hex (1 to 100 words)

FINS unit address of relaying Communications Unit.
- CPU Bus Unit: 10 to 1F hex (unit number + 10 hex)
- Special I/O Unit: 20 to 7F hex (unit number + 20 hex)

Port number of the communications port (logical port) for the network instruction:
0 to 7 hex (F hex: Automatic allocation)

Response monitoring time
0001 to FFFF hex (0.1 to 6553.5 s)
0000 hex: 2 s (default setting)

Explicit message format
0000 hex: DeviceNet
(same as using the 2801 FINS command)

Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK*	OK	OK	---	---	---	OK	---	---	---	---
D																		
C																		

* The D bank and higher of the EM Area can be specified only for CJ2 CPU Units. The D bank and higher cannot be specified in the program in a CS/CJ-series CPU Unit even if the source is a CJ2 CPU Unit.

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. • OFF in all other cases.

Related Auxiliary Area Words and Bits

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 *About Explicit Message Instructions*.

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.

Function

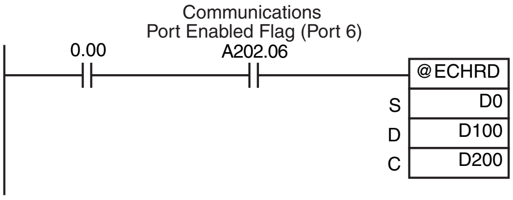
Reads the specified number of words from the first read word (specified in S) in the remote CPU Unit with the node address specified in C, and stores the data in the local CPU Unit memory words beginning at D.

Hint

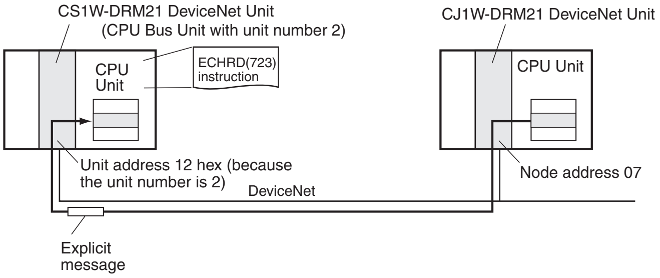
- ECHRD(723) sends an explicit message with the Service Code 1C hex (Byte Data Read).

Example Programming

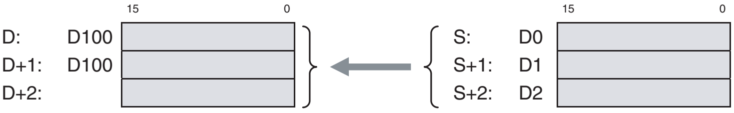
In this example, ECHRD(723) is used to read the I/O memory of the CJ-series CPU Unit on the DeviceNet network, and store the data in the I/O memory of the local CPU Unit.



When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) are ON, ECHRD(723) reads D0 to D2 from the I/O memory of the CJ-series CPU Unit with node address 07 on the DeviceNet Network and stores the data in D100 to D102 of the local CPU Unit.

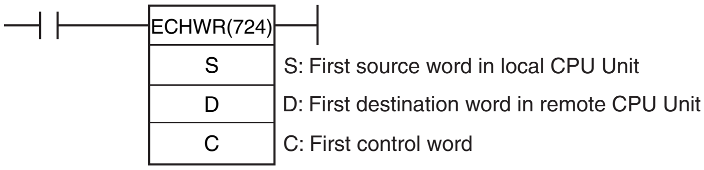


		15	8	7	0	
C:	D200	0	0	0	7	Node address of remote CPU Unit to be read = 07 hex (node 07)
C+1:	D201	0	0	0	3	Read data size (number of words) = 3 hex
C+2:	D202	0	6	1	2	Communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+3:	D203	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+4:	D204	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)



ECHWR

Instruction	Mnemonic	Variations	Function code	Function
EXPLICIT WORD WRITE	ECHWR	@ECHWR	724	Writes data from the local CPU Unit to another CPU Unit in the network. (The remote CPU Unit must support explicit messages.)

	ECHWR
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word in local CPU Unit	UINT	1
D	First destination word in remote CPU Unit	UINT	1
C	First control word	WORD	5

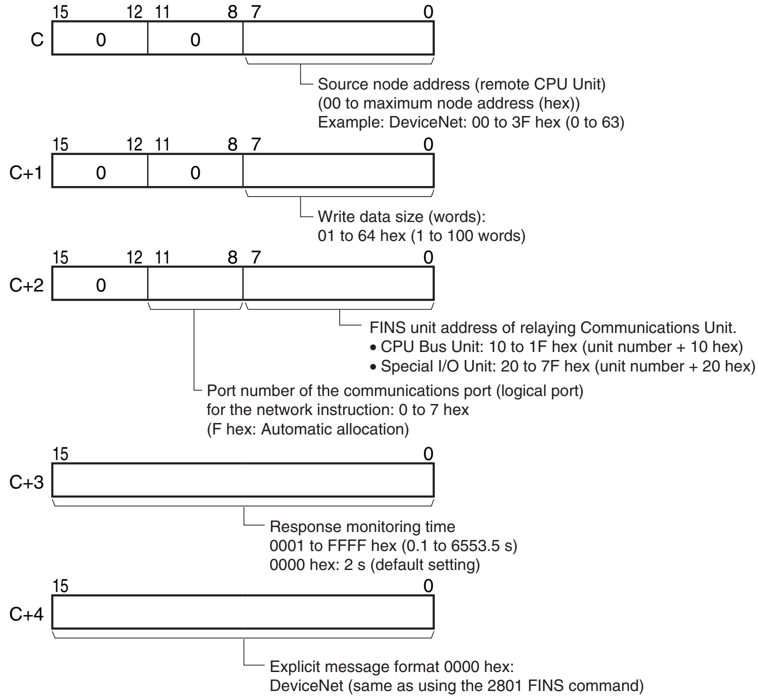
S: First Source Word in Local CPU Unit

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



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S D C	OK	OK	OK	OK	OK	OK	OK	OK*	OK	OK	---	---	---	OK	---	---	---	---

* The D bank of the EM Area can be specified only for CJ2 CPU Units. The D bank and higher cannot be specified in the program in a CS/CJ-series CPU Unit even if the destination is a CJ2 CPU Unit.

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. • OFF in all other cases.

Related Auxiliary Area Words and Bits

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 *About Explicit Message Instructions*.

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.

Function

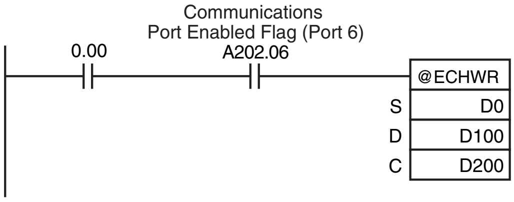
Writes the specified number of words beginning at S from the local CPU Unit to the write destination beginning at D in the remote CPU Unit with the node address specified in C.

Hint

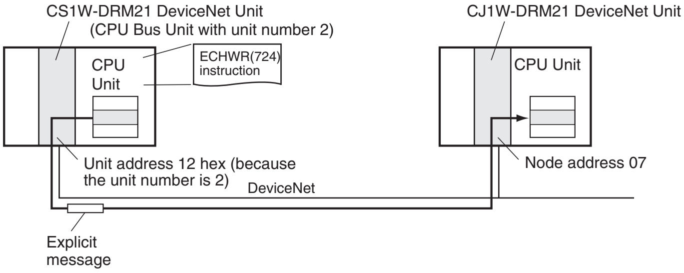
- ECHWR(724) sends an explicit message with the Service Code 1E hex (Byte Data Write).

Example Programming

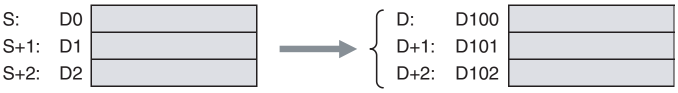
In this example, ECHWR(724) is used to write data from the I/O memory of the local CPU Unit to the I/O memory of a CJ-series CPU Unit on the DeviceNet network.



When CIO 0.00 and A202.06 (the Communications Port Enabled Flag for port 06) 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 07 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)



File Memory Instructions

File Memory Instructions

User Instructions for File Memory Operations

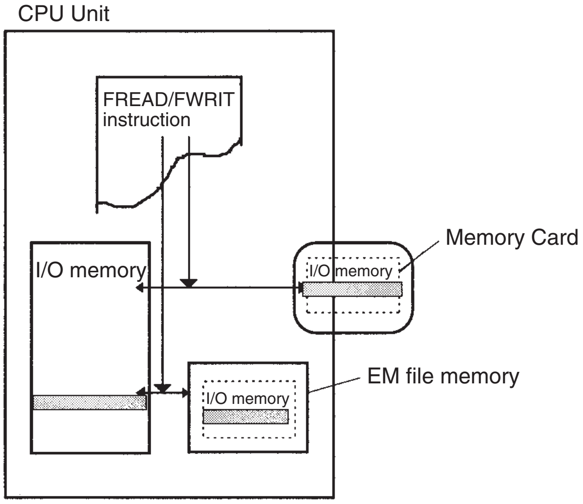
● Overview

- The FWRIT(701) (WRITE DATA FILE)/TWRIT(704) (WRITE TEXT FILE) instruction can be used to create a data file containing the specified I/O memory data in a Memory Card or EM file memory. It can also add to or overwrite from any point in existing files.
- The FREAD(700) (READ DATA FILE) instruction will read I/O memory data from a specified location from a data file in a Memory Card or EM file memory and write it to the specified portion of I/O memory. It can read from any point in the specified file.

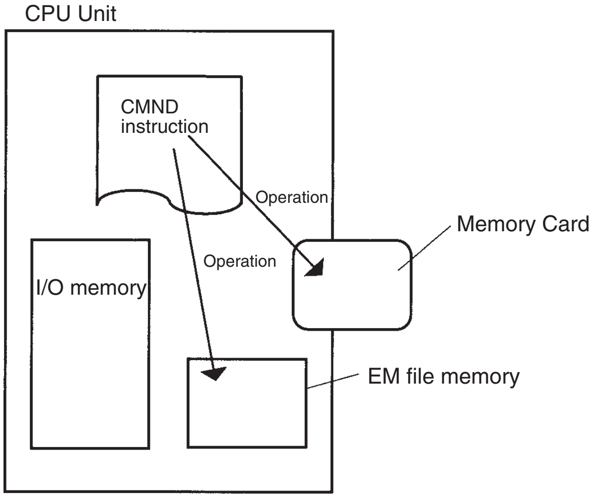
Note These instructions do not transfer the specified file, but rather the specified amount of data beginning at the specified start position in the file.

- The CMND(490) (DELIVER COMMAND) instruction can be executed to issue a FINS command to the CPU Unit itself to perform file operations. File operations such as file formatting, deletion, copying, and renaming can be performed on files in the Memory Card or EM file memory.

FREAD(700)/FWRIT(701):
Transfers between I/O memory and file memory



CMND(490):
File memory operations



● File Memory Instructions
READ(700)/FWRIT(701) Instructions

Name	Mnemonic	Description
READ DATA FILE	FREAD(700)	Reads specified data file data or data elements to specified I/O memory.
WRITE DATA FILE	FWRIT(701)	Uses specified I/O memory area data to create a specified data file.
WRITE TEXT FILE (See note.)	TWRIT(704)	Creates a text file from ASCII data in the specified I/O memory data area location.

The operands of the FREAD(700) and FWRIT(701) instructions determine whether binary data (.IOM files), non-delimited or tab-delimited text files (.TXT files), or comma-delimited text files (.CSV) are handled. The operand settings are shown below.

FREAD(700) and FWRIT(701) Operands

Bits in C	Settings	Programming Device limitations
12 to 15	Data type 0: Binary (.IOM) 1: Non-delimited words (.TXT) 2: Non-delimited double-words (.TXT) 3: Comma-delimited words (.CSV) 4: Comma-delimited double-words (.CSV) 5: Tab-delimited words (.TXT) 6: Tab-delimited double-words (.TXT)	0 and 6 hex
08 to 11	Carriage returns 0: No returns 8: Return every 10 fields 9: Return every 1 field A: Return every 2 fields B: Return every 4 fields C: Return every 5 fields D: Return every 16 fields	0 hex or to between 8 and D hex

Hint

Files created in memory using the FWRIT(701) instruction will be dated using the data on the clock built into the CPU Unit.

You cannot execute more than one FREAD(700) or FWRIT(701) instruction, execute a CMND(490) instruction for the local CPU Unit, replace the program using an Auxiliary Area bit, or perform a simple backup at the same time. Perform exclusive control in the program using the File Memory Operation Flag (A343.13).

If an error occurs during FREAD(700) because illegal data is read (data that is not hexadecimal or not 8-digit or 8-digit data), reading will be stopped, but the data read to that point will remain in memory. The File Read Error Flag (A343.10) will turn ON (but the Error Flag (P_ER) will not turn ON.

Related Auxiliary Bits/Words

Name	Address	Operation
Memory Card Type	A343.00 to A343.02	Indicates the type of Memory Card, if any, that is installed.
EM File Memory Format Error Flag	A343.06	ON when a format error occurs in the first EM bank allocated for file memory. OFF when formatting is completed normally.
Memory Card Format Error Flag	A343.07	ON when the Memory Card is not formatted or a formatting error has occurred.
File Write Error Flag	A343.08	ON when an error occurred when writing to the file.
File Write Impossible Flag	A343.09	ON when the data couldn't be written because the file was write-protected or there was insufficient free memory.
File Read Error Flag	A343.10	ON when a file could not be read because its data was corrupted or if it contains the wrong data type.
No File Flag	A343.11	ON when data could not be read because the specified file doesn't exist.
File Memory Operation Flag	A343.13	ON for any file memory operation. OFF when no file memory operation is in progress. • Starting Memory Card detection. • The CPU Unit is processing a FINS command sent to itself using CMND(490). • FREAD(700) or FWRIT(701) is being executed. • The program is being overwritten using an Auxiliary Area control bit. • A simple backup operation is being performed.
Accessing File Flag	A343.14	ON when file data is actually being accessed.
Memory Card Detected Flag	A343.15	ON when a Memory Card has been detected. OFF when a Memory Card is not detected.
Number of Items to Transfer	A346 to A347	These words indicate the number of words or fields remaining to be transferred (32 bits). When a binary (.IOM) file is being transferred, this number is decremented each time a word is read. When a text or CSV file is being transferred, this number is decremented each time a field is transferred.

● CMND(490): DELIVER COMMAND Instruction

Name	Mnemonic	Operation
DELIVER COMMAND	CMND (490)	CMND(490) can be used to issue a FINS command to the local CPU Unit itself to perform file memory operations such as formatting or deleting files.

Note Make the following settings in CMND(490)'s control words:
Destination network address: 00 hex (local network), Serial port: 0 hex (not used), Destination unit address: 00 hex (CPU Unit), Destination node address: 00 hex (local node), Retries: 0 hex (Set to 0 because retrying is disabled.)

FINS Commands Related to File Memory

Note There are other FINS commands related to file memory that are not shown in the following table that can be executed. Refer to the Communications Command Reference Manual (W342) for details on FINS commands.

CMND(490) cannot be executed to the local CPU Unit if another CMND(490) or file memory instruction is being executed to another CPU Unit, A File Memory Instruction is being executed, the program is being replaced by an Auxiliary Area control bit operation, or a simple backup operation is being executed. Be sure to include the File Memory Operation Flag (A34313) as a normally closed condition to prevent CMND(490) from being executed while another file memory operation is in progress.

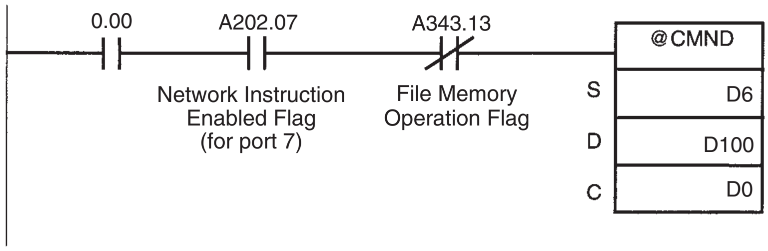
If CMND(490) cannot be executed for the local CPU Unit, the Error Flag will be turned ON.

Related Auxiliary Bits/Words

Name	Address	Operation
File Memory Operation Flag	A343.13	ON for any file memory operation. OFF when no file memory operation is in progress. • Starting Memory Card detection. • The CPU Unit is processing a FINS command sent to itself using CMND(490). • FREAD(700) or FWRIT(701) is being executed. • The program is being overwritten using an Auxiliary Area control bit. • A simple backup operation is being performed.
Memory Card Detected Flag	A343.15	ON when a Memory Card has been detected. OFF when a Memory Card is not detected.

● Example Program

Example: The following example shows how to use CMND(490) to create a subdirectory in the Memory Card.



When 0 and A202.07 are ON and A343.13 is OFF, CMND(490) issues FINS command 2215 (CREATE/DELETE SUBDIRECTORY) is sent to the local CPU Unit and the response is stored in D100 and D101.

In this case, the FINS command creates a subdirectory named "CS1" within the OMRON directory in the CPU Unit's Memory Card. The response is composed of the 2-byte command code (2215) and the 2-byte response code.

		15	8	7	0	
S:	D6	2	2	1	5	Command code: 2215 Hex (CREATE/DELETE SUBDIRECTORY)
S+1:	D7	8	0	0	0	Disk number: 8000 Hex (Memory Card)
S+2:	D8	0	0	0	0	Parameter: 0000 Hex (Create subdirectory.)
S+3:	D9	4	3	5	3	Subdirectory name: CS1□□□□□.□□□ (□: a space)
S+4:	D10	3	1	2	0	
S+5:	D11	2	0	2	0	
S+6:	D12	2	0	2	0	
S+7:	D13	2	E	2	0	
S+8:	D14	2	0	2	0	
S+9:	D15	0	0	0	6	Directory length: 0006 Hex (6 characters)
S+10:	D16	5	C	4	F	Directory path: \OMRON
S+11:	D17	4	D	5	2	
S+12:	D18	4	F	4	E	

		15	8	7	0	
C:	D0	0	0	1	A	Number of bytes of command data: 001A Hex (26 bytes)
C+1:	D1	0	0	0	4	Number of bytes of response data: 0004 Hex (4 bytes)
C+2:	D2	0	0	0	0	Destination address: 0000 Hex (local network) 00 Hex (local node) and 00 Hex (CPU Unit)
C+3:	D3	0	0	0	0	
C+4:	D4	0	7	0	0	Response requested, communications port 7, 0 retries
C+5:	D5	0	0	0	0	Response monitor time: FFFF Hex (6,553.5 s)

Hint

There are other FINS commands that can be sent to the local PLC in addition to the ones related to file memory operations that are listed in the table above. The File Memory Operation Flag must be used to prevent simultaneous execution of these other FINS commands, too.

● Procedure

Reading/Writing a Data File on a Memory Card or Performing a File Memory Operation

1. Place a formatted Memory Card into the CPU Unit.
2. Using the FWRIT(701) instruction, name a file in the specified I/O memory area and save the data in the Memory Card.

Note To use the TXT or CSV data that was written to a Memory Card in spreadsheet software on a computer, insert the Memory Card into an HMC-AP001 Memory Card Adapter mounted in a PC card slot of the computer and used Windows to read the data.

Use the FREAD(700) instruction to read the data file from the Memory Card to I/O memory in the CPU Unit and then manipulate the file in the Memory Card by using the CMND(490) to send a command to the local CPU Unit.

Reading/Writing a Data File in EM File Memory or Performing Another File Memory Operation

1. Convert all banks from the specified bank of the EM Area to file memory using the settings in the PLC Setup.

2. Initialize EM file memory from the CX-Programmer.

3. Using the FWRIT(701) instruction, name a file in the specified I/O memory area and save the data in the EM file memory.
   Use the FREAD(700) instruction to read the data file from the EM file memory to I/O memory in the CPU Unit and then manipulate the file in the Memory Card by using the CMND(490) to send a command to the local CPU Unit.

Precautions when Using Memory Cards

Confirm the following items before using a Memory Card.

(1) Format

Memory Cards are formatted before shipping. There is no need to format them after purchase. To format them once they have been used, always do so in the CPU Unit using the CX-Programmer or a Programming Console.

If a Memory Card is formatted directly in a notebook computer or other computer, the CPU Unit may not recognize the Memory Card. If this occurs, you will not be able to use the Memory Card even if it is reformatted in the CPU Unit.

(2) Number of Files in Root Directory

There is a limit to the number of files that can be placed in the root directory of a Memory Card (just as there is a limit for a hard disk). Although the limit depends on the type and format of the Memory Card, it will be between 128 and 512 files. When using applications that write log files or other files at a specific interval, write the files to a subdirectory rather than to the root directory.

Subdirectories can be created on a computer or by using the CMND(490) instruction. Refer to 3-26-6 DELIVER COMMAND: CMND(490) for a specific example using CMND(490).

(3) Number of Writes

Generally speaking, there is no limit to the number of write operations that can be performed for a flash memory. For the Memory Cards, however, a limit of 100,000 write operations has been set for warranty purposes. For example, if the Memory Card is written to every 10 minutes, over 100,000 write operations will be performed within 2 years.

(4) Minimum File Size

If many small files, such as ones containing only a few words of DM Area data, are stored on the Memory Card, it will not be possible to use the complete capacity of the Memory Card. For example, if a Memory Card with an allocation unit size of 4,096 bytes is used, at least 4,096 bytes of memory will be used for each file regardless of how small the file is. If you save 10 words of DM Area data to the Memory Card, 4,096 bytes of memory will be used even though the actual file size is only 68 bytes. Using files of such a small size greatly reduces the utility rate of the Memory Card. If the allocation unit size is reduced to increase the utility rate, however, the access speed will be reduced.
The allocation unit size of the Memory Card can be checked from a DOS prompt using CHKDSK. The specific procedure is omitted here. Refer to general computer references for more information on allocation unit sizes.

(5) Memory Card Access Precautions

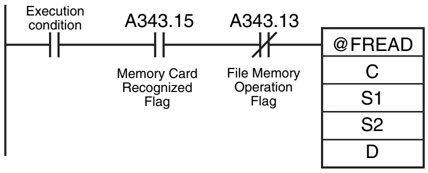
When the PLC is accessing the Memory Card, the BUSY indicator will light on the CPU Unit. Observe the following precautions.

1. Never turn OFF the power supply to the CPU Unit when the BUSY indicator is lit. The Memory Card may become unusable if this is done.

2. Never remove the Memory Card from the CPU Unit when the BUSY indicator is lit. Press the Memory Card power OFF button and wait for the BUSY indicator to go out before removing the Memory Card. The Memory Card may become unusable if this is not done.

3. Insert the Memory Card with the label facing to the right. Do not attempt to insert it in any other orientation. The Memory Card or CPU Unit may be damaged.

4. A few seconds will be required for the CPU Unit to recognize the Memory Card after it is inserted. When accessing a Memory Card immediately after turning ON the power supply or inserting the Memory Card, program an NC condition for the Memory Card Recognized Flag (A343.15) as an input condition, as shown below.



File Memory Instructions

● FWRIT(701)

FWRIT(701) creates a data file containing the specified data from I/O memory. The file format can be either binary or CSV. FWRIT(701) can also be used to add to an existing file or overwrite an existing file from a specified position.

● FREAD(700)

FREAD(700) reads the contents of a data file and stores it in the specified area of I/O memory. The file format can be either binary or CSV. FREAD(700) can also be used to read data from a specified position in a file.

● TWRIT(704)

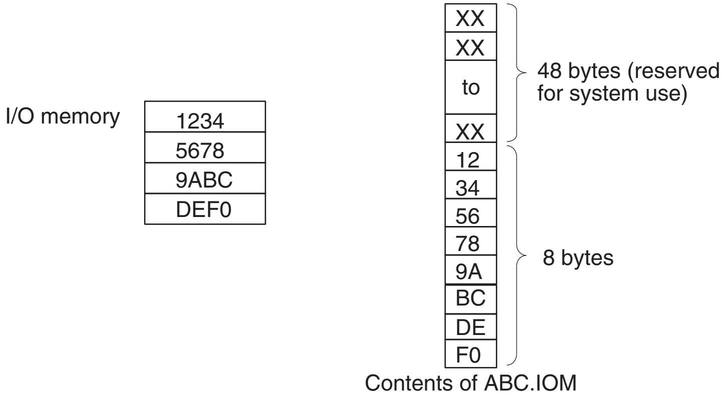
TWRIT(704) creates a text file containing ASCII data stored in I/O memory. TWRIT(704) can also be used to add to an existing file or overwrite an existing file.

● CMND(490)

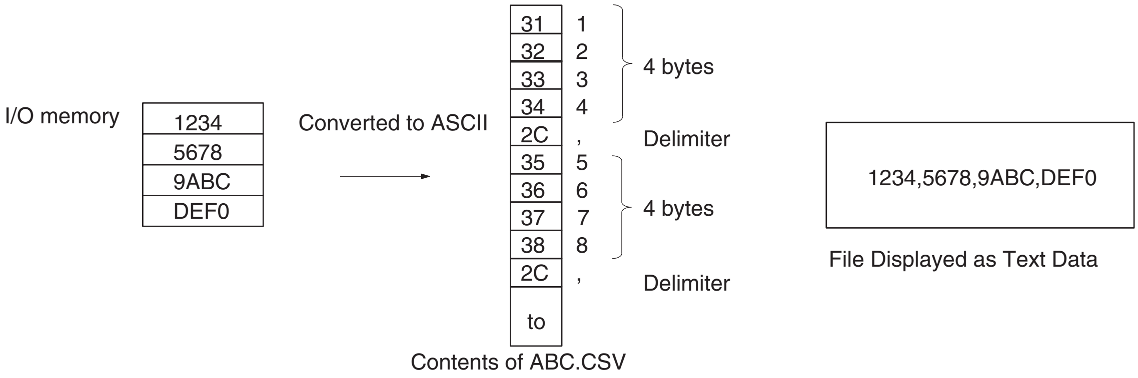
CMND(490) can be used to format files, delete files, copy files, and change file names by sending FINS commands for Memory Card operations.

Data structure of Files

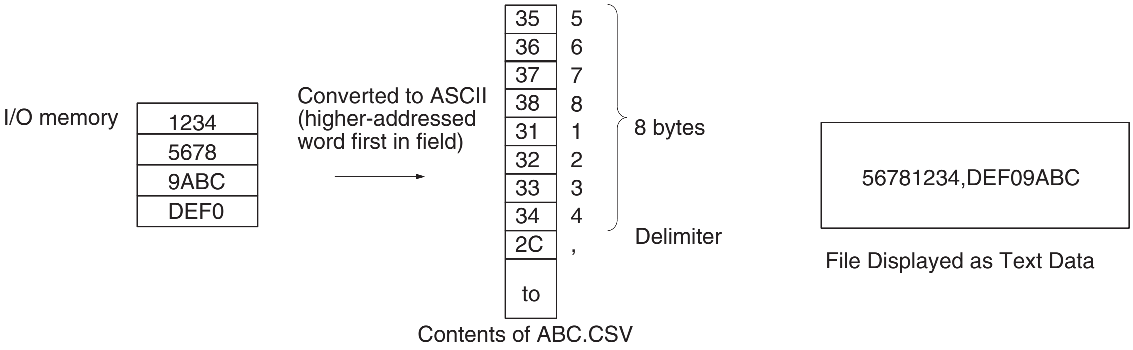
For binary format (.IOM), the data will be as follows when 1234 hex, 5678 hex, 9ABC hex, and DEF0 hex are stored in the file ABC.IOM (although the user does not normally need to be concerned with this structure):



For word CSV format (.CSV), the data will be as follows when 1234 hex, 5678 hex, 9ABC hex, and DEF0 hex are stored in the file ABC.CSV (the basic structure would be the same for text data (.TXT):



For long-word CSV format (.CSV), the data will be as follows when 1234 hex, 5678 hex, 9ABC hex, and DEF0 hex are stored in the file ABC.CSV (the basic structure would be the same for text data (.TXT):



Related Auxiliary Area Words and Bits

● Memory Card Detection

Name	Address	Operation
Memory Card Type	A343.00 to A343.02	Contains a binary number indicating the type of Memory Card, if any, that is installed. (0: None, 4: Flash ROM)
Memory Card Format Error Flag	A343.07	ON when the Memory Card is not formatted or a formatting error has occurred.
Memory Card Detected Flag	A343.15	ON when a Memory Card has been detected. OFF when a Memory Card is not detected.

● Instruction-related Words and Bits

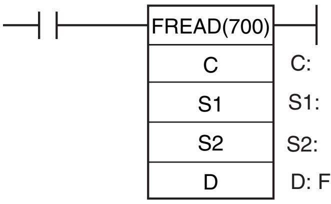
Name	Address	Operation
File Write Error Flag	A343.08	ON when an error occurred when writing to the file. ON when the file being written is write-protected.
File Write Impossible Flag	A343.09	ON when the data could not be written because there was insufficient free memory.
File Read Error Flag	A343.10	ON when a file could not be read because its data was corrupted or if it contains the wrong data type.
File Missing Flag	A343.11	ON when data could not be read because the specified file does not exist.
File Memory Operation Flag	A343.13	ON for any of the following: • The CPU Unit has sent a FINS command to itself using CMND(490). • FREAD(700) or FWRIT(701) are being executed. • The program is being overwritten using a control bit in memory. • A simple backup operation is being performed.
Accessing File Flag	A343.14	ON when file data is actually being accessed. Use this flag as an execution condition to prevent a file memory instruction from being executed while another is in progress.
Number of Data to Transfer	A346 to A347	The contents of these words indicate the status of data file transfers. When an FREAD(700) or FWRIT(701) instruction is executed, the number of words or fields to be transferred is written to these words. The value is decremented by 1 as each word or field is transferred. A346 contains the rightmost 16 bits and A347 contains the leftmost 16 bits of the 32-bit binary value.

● EM File Memory-related Words and Bits

Name	Address	Operation
EM File Memory Format Error Flag	A343.06	ON when there is a format error in the starting bank of EM file memory.
EM File Format Starting Bank	A344	Contains the starting bank number of the EM Area that has been formatted for use as EM file memory. Contains FFFF when none of the EM Area has been formatted.

FREAD

Instruction	Mnemonic	Variations	Function code	Function
READ DATA FILE	FREAD	@FREAD	700	Reads the specified data or amount of data from the specified data file in file memory to the specified data area in the CPU Unit.

	FREAD
Symbol	 C: Control word S1: Number of words and First source word S2: Filename D: First destination word

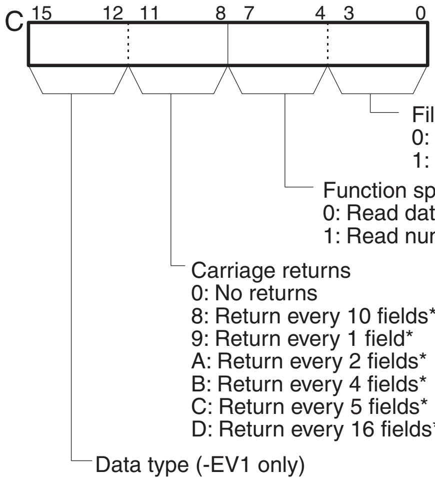
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	UINT	Variable
S1	Number of words and First source word	LWORD	4
S2	Filename	UINT	Variable
D	First destination word	UINT	Variable

C: Control Word



File memory specifier
0: Memory Card
1: EM file memory

Function specifier
0: Read data.
1: Read number of words.

Carriage returns
0: No returns
8: Return every 10 fields*
9: Return every 1 field*
A: Return every 2 fields*
B: Return every 4 fields*
C: Return every 5 fields*
D: Return every 16 fields*

Data type (-EV1 only)
0: Binary (extension: .IOM, words/field: NA)
1: Non-delimited words (extension: .TXT, words/field: 1)*
2: Non-delimited double-words (extension: .TXT., words/field: 2)*
3: Comma-delimited words (extension: .CSV, words/field: 1)*
4: Comma-delimited double-words (extension: .CSV, words/field: 2)*
5: Tab-delimited words (extension: .TXT, words/field: 1)*
6: Tab-delimited double-words (extension: .,TXT words/field: 2)*

*: Cannot be set for CS-series CS1 CPU Units prior to V1□.

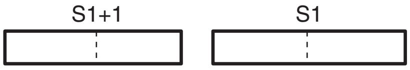
Note 1 Each field will contain 1 word of I/O memory for the word data types and 2 words of I/O memory for the double-word data types.

2 When reading data with carriage returns, bits 00 to 11 of C must be set to between 8 and D hex.

3 With double-words, the first word of data is stored in the higher memory address, e.g., 12345678 would be stored with 1234 in D1 and 5678 in D0.

S1: Number of read words, starting read position

- S1 and S1+1: Number of Read Items

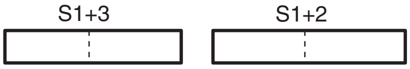


S1+1 contains the leftmost 4 digits and
S1 contains the rightmost 4 digits.

S1+1 is the leftmost 4 digits, S1 is the rightmost 4 digits

Data type	Bits 12 to 15 of C	Contents of S1 and S1+1
Binary	0 hex (binary)	Number of words to read from file memory. 00000000 to 3FFFFFFF hex
Word	1 hex (non-delimited), 3 hex (comma-delimited), or 5 hex (tab-delimited)	Number of fields to read from file memory, i.e., the number of words to read from file memory. 00000000 to 1FFFFFFF hex
Double-word	2 hex (non-delimited), 4 hex (comma-delimited), or 6 hex (tab-delimited)	Number of fields to read from file memory, i.e., half the number of words to read from file memory. 00000000 to 0FFFFFFF hex

- S1+2 and S1+3: First Source Word
  The 8-digit hexadecimal value in S1+2 and S1+3 specifies the starting read word from the beginning of the file.



S1+3 contains the leftmost 4 digits and
S1+2 contains the rightmost 4 digits.

S1+3 is the leftmost 4 digits, S1+2 is the rightmost 4 digits

Data type	Bits 12 to 15 of C	Contents of S1+2 and S1+3
Binary	0 hex (binary)	The word at which to begin reading from the beginning of file memory. 00000000 to 3FFFFFFF hex
Word	1 hex (non-delimited), 3 hex (comma-delimited), or 5 hex (tab-delimited)	The field at which to begin reading from the beginning of file memory, i.e., the number of words from the beginning. 00000000 to 1FFFFFFF hex
Double-word	2 hex (non-delimited), 4 hex (comma-delimited), or 6 hex (tab-delimited)	The field at which to begin reading from the beginning of file memory, i.e., half the number of words from the beginning. 00000000 to 0FFFFFFF hex

Note
- S1+2 and S1+3 are used only for text and CVS data with no carriage returns (i.e., bits 08 to 11 of C set to 0 hex) or for binary data. Always set S1+2 and S1+3 to 00000000 hex when reading data with carriage returns (i.e., bits 08 to 11 of C set to between 8 and D hex).
- S1 to S1+3 must be in the same data area. S1 to S1+3 are used only when reading data.

S2: Filename

- S2 is the starting address of the words containing the absolute path and filename in ASCII. Use ASCII a to z, A to Z, and 0 to 9.
- The full path name to the directory containing the data file can be up to 65 characters long, including the starting slash (ASCII 5C). The filename can be up to 8 characters long, but null characters (ASCII 00) are not allowed in the filename because the null character is used to mark the end of the character string. The extension is specified by C, and thus is not included in the file name.
- Separate the directory name and file name with ¥ (#5C).

S2	F1	F2
S2+1	F3	F4
⋮	⋮	⋮
S2+38	F73	F74

Store the character string beginning with the leftmost byte in S2.
The entire pathname and filename can be up to 74 characters (bytes) long, including the initial slash character and ending null character.

D: First Destination Word

- When data is being read, D specifies the starting address where the data read from file memory will be stored.
- When the number of words of data is being read, the number of words is written to D and D+1 in 8-digit hexadecimal (00000000 to 7FFFFFF). D contains the rightmost 4 digits and D+1 contains the leftmost 4 digits.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	OK	---	---	---	---
S1, S2									OK	OK	---							
D																		

Flags

Name	Label	Operation
Error Flag	ER	• ON if the file memory specified in C does not exist. • ON if the settings in C are not within the specified range. • ON if the filename specified in S2 does not satisfy the required conditions. • ON if the File Memory Operation Flag was ON. • ON if a constant was not specified for C (only for CS-series CS1 CPU Units prior to V1□). • ON if data specified for S1 is out of range (all CPU Units except for CS-series CS1 CPU Units prior to V1□). • ON if an illegal area is specified for D. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.

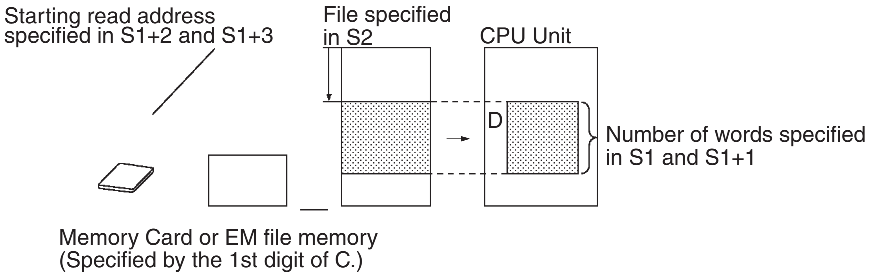
Related Auxiliary Area Flags and Words

Name	Address	Operation
Memory Card Type	A343.00 to A343.02	Contains a binary number indicating the type of Memory Card, if any, that is installed. (0: None, 4: Flash ROM)
Memory Card Format Error Flag	A343.07	ON when the Memory Card is not formatted or a formatting error has occurred.
File Read Error Flag	A343.10	ON when a file could not be read because its data was corrupted or if it contains the wrong data type.
File Missing Flag	A343.11	ON when data could not be read because the specified file does not exist.
File Memory Operation Flag	A343.13	ON for any of the following: • The CPU Unit has sent a FINS command to itself using CMND(490). • FREAD(700) or FWRIT(701) are being executed. • The program is being overwritten using a control bit in memory. • A simple backup operation is being performed. • Use this flag as an execution condition to prevent a file memory instruction from being executed while another is in progress.
Accessing File Flag	A343.14	ON when file data is actually being accessed.
Memory Card Detected Flag	A343.15	ON when a Memory Card has been detected.
EM File Memory Format Error Flag	A343.06	ON when there is a format error in the starting bank of EM file memory.
Number of Data to Transfer	A346 to A347	The contents of these words indicate the status of data file transfers. When an FREAD(700) or FWRIT(701) instruction is executed, the number of words or fields to be transferred is written to these words. The value is decremented by 1 as each word or field is transferred. A346 contains the rightmost 16 bits and A347 contains the leftmost 16 bits of the 32-bit binary value.

Function

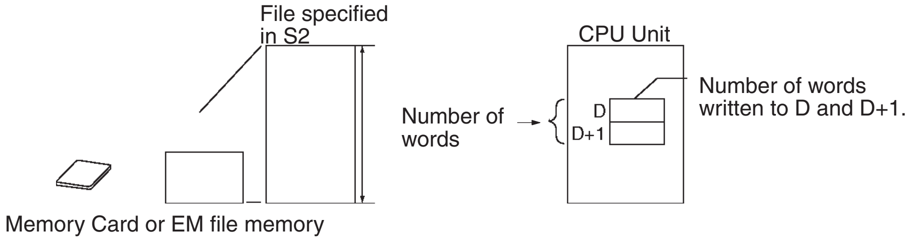
● Reading Data (Third Digit of C = 0)

FREAD(700) reads the number of words or fields specified in S1 and S1+1 from the file specified in S2 (with filename extension .IOM, .TXT, or .CSV) beginning at the address specified in S1+2 and S1+3. The data is then written to RAM beginning at the word specified in D.



● Reading Number of Words of Data (Third Digit of C=1)

FREAD(700) finds the number of words in the file specified in S2 (with filename extension .IOM) and writes that 8-digit hexadecimal value to D and D+1.



Hint

- When another file memory related operation (file memory format, file copy, file delete, etc.) is executed from the ladder program, send the file memory related FINS command to the local CPU Unit with a CMND(490) instruction.
- Write the path name and filename in ASCII beginning with the leftmost byte of S2, as shown in the following example for \ABC\XYZ.IOM. (The .IOM extension is added automatically.)

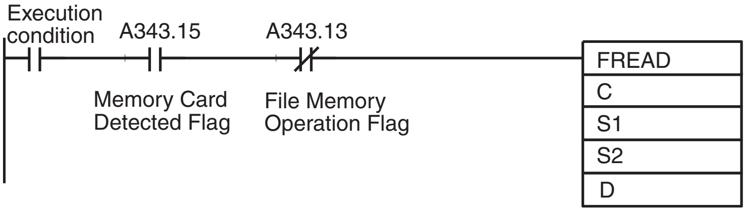
S2	"\"	"A"
S2+1	"B"	"C"
S2+2	"\"	"X"
S2+3	"Y"	"Z"
S2+4	NUL	

S2	5C	41
S2+1	42	43
S2+2	5C	58
S2+3	59	5A
S2+4	00	

Precaution

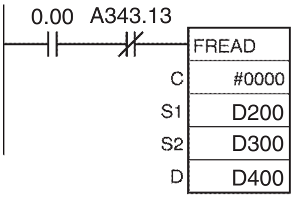
- During normal instruction processing, FREAD(700) is used only to start reading file memory. The instruction execution times given toward the end of this manual are thus the times required to start reading, not to complete it. Actual reading (transfer) is performed by the file access processing in peripheral servicing. Therefore, once FREAD(700) has been executed, reading is continuously executed even if the execution condition is OFF in following cycles.
  The time required to complete data transfer for FREAD(700) will depend on the amount of data being transferred, the service time allocated to file access processing, and other conditions. As a guideline, the transfer times for a cycle time of 10 ms for a file in the root directory with the default service time settings will be 0.92 s for 1,024 words and 4.64 s for 9,999 words.
- When transfer has been completed, the File Memory Operation Flag (A343.13) will turn OFF. This flag can be used for exclusive control of file memory instructions.
- Data is stored in order by absolute internal memory addresses, so the output data will overwrite data in the next data area if it exceeds the capacity of the data area specified in D. See Precautions for more details.
- When FREAD(700) is executed, the number of words (or fields) specified in S1 and S1+1 is written to A346 and A347 (Number of Data to Transfer) and this value is decremented by 1 as each word or field is transferred. The content of these words can be checked to verify that the expected number of words or fields were transferred.
- If the specified number of words or fields exceeds the number of words in the data file, the data in the file will be transferred normally and no error will occur.
- If the specified starting word exceeds the number of words in the data file, the File Read Error Flag (A343.10) will be turned ON and the file data will not be read.
- If the specified file or directory does not exist, the File Missing Flag (A343.11) will be turned ON and the file data will not be read.
- The File Memory Operation Flag (A343.13) will be turned ON when FREAD(700) is executed. An error will occur and the instruction will not be executed if A34313 is already ON.
- The File Read Error Flag (A343.10) will be turned ON and the instruction will not be executed if the specified file contains the wrong data type or the file data is corrupted. For text or CSV files, the character code must be hexadecimal data and delimiters must be every 4 digits for word data and every 8 digits for double-word data. Data will be read up to the point where an illegal character is detected.

- A few seconds is required for the CPU Unit to detect a Memory Card after it has been inserted. If a Memory Card is going to be accessed soon after power is turned ON or after a Memory Card is inserted, use the Memory Card Detected Flag (A343.15) in a NO input condition as shown below to be sure that the Memory Card has been detected.



Example Programming

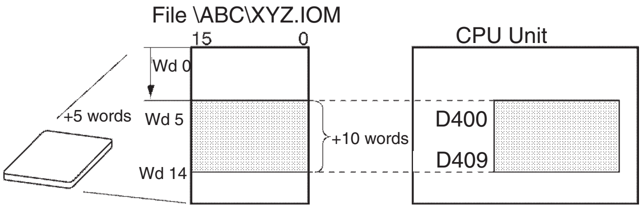
When CIO 0.00 turns ON in the following example, FREAD(700) reads 10 words of data from file \ABC\XYZ.IOM starting with the beginning of the file + 5 words and outputs these 10 words to D400 through D409.



C: # 0 0 0 0 File memory: Memory Card
Function: Read data

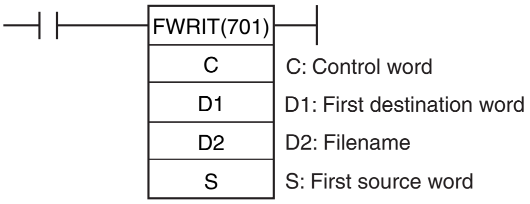
S1: D200	0 0 0 A	Number of words to read: 10 words
S1+1: D201	0 0 0 0	
S1+2: D202	0 0 0 5	Starting word: Beginning of file+5 words
S1+3: D203	0 0 0 0	

S2: D300	5 C 4 1	Directory name: \ABC
S2+1: D301	4 2 4 3	Filename: XYZ
S2+2: D302	5 C 5 8	
S2+3: D303	5 9 5 A	
S2+3: D303	0 0 Ignored	



FWRIT

Instruction	Mnemonic	Variations	Function code	Function
WRITE DATA FILE	FWRIT	@FWRIT	701	Overwrites or appends data in the specified data file in file memory with the specified data from the data area in the CPU Unit.

	FWRIT
Symbol	

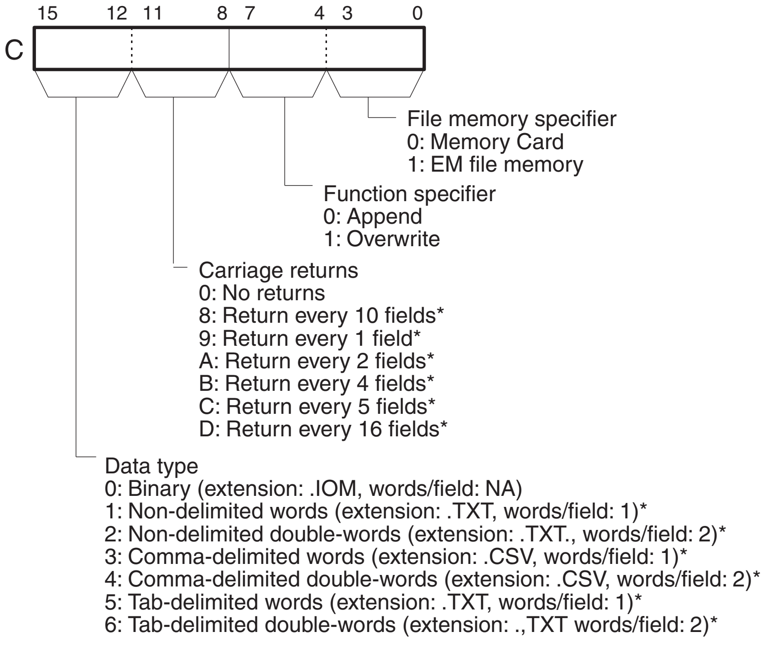
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	UINT	Variable
D1	First destination word	LWORD	4
D2	Filename	UINT	Variable
S	First source word	UINT	Variable

C: Control Word



File memory specifier
0: Memory Card
1: EM file memory

Function specifier
0: Append
1: Overwrite

Carriage returns
0: No returns
8: Return every 10 fields*
9: Return every 1 field*
A: Return every 2 fields*
B: Return every 4 fields*
C: Return every 5 fields*
D: Return every 16 fields*

Data type
0: Binary (extension: .IOM, words/field: NA)
1: Non-delimited words (extension: .TXT, words/field: 1)*
2: Non-delimited double-words (extension: .TXT., words/field: 2)*
3: Comma-delimited words (extension: .CSV, words/field: 1)*
4: Comma-delimited double-words (extension: .CSV, words/field: 2)*
5: Tab-delimited words (extension: .TXT, words/field: 1)*
6: Tab-delimited double-words (extension: .,TXT words/field: 2)*

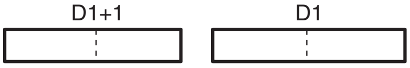
*: Cannot be set for CS-series CS1 CPU Units prior to V1□.

Note 1 Each field will contain 1 word of I/O memory for the word data types and 2 words of I/O memory for the double-word data types.

2 With double-words, the first word of data is read from the higher memory address, e.g., 12345678 would be written with 1234 from D00001 and 5678 from D00000.

D1: Number of write words, starting write position

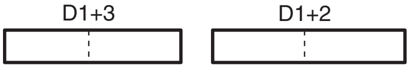
- D1 and D1+1: Number of Write Items



D1+1 contains the leftmost 4 digits and
D1 contains the rightmost 4 digits.

Data type	Bits 12 to 15 of C	Contents of D1 and D1+1
Binary	0 hex (binary)	Number of words to write from file memory. 00000000 to 3FFFFFFF hex
Word	1 hex (non-delimited), 3 hex (comma-delimited), or 5 hex (tab-delimited)	Number of fields to write from file memory, i.e., the number of words to write from file memory. 00000000 to 1FFFFFFF hex
Double-word	2 hex (non-delimited), 4 hex (comma-delimited), or 6 hex (tab-delimited)	Number of fields to write from file memory, i.e., half the number of words to write from file memory. 00000000 to 0FFFFFFF hex

- D1+2 and D1+3: First Destination Word
  The 8-digit hexadecimal value in D1+2 and D1+3 specifies the starting write word from the beginning of the file.



D1+3 contains the leftmost 4 digits and
D1+2 contains the rightmost 4 digits.

Data type	Bits 12 to 15 of C	Contents of D1+2 and D1+3
Binary	0 hex (binary)	The word at which to begin writing from the beginning of file memory. 00000000 to 3FFFFFFF hex
Word	1 hex (non-delimited), 3 hex (comma-delimited), or 5 hex (tab-delimited)	The field at which to begin writing from the beginning of file memory, i.e., the number of words from the beginning. 00000000 to 1FFFFFFF hex
Double-word	2 hex (non-delimited), 4 hex (comma-delimited), or 6 hex (tab-delimited)	The field at which to begin writing from the beginning of file memory, i.e., half the number of words from the beginning. 00000000 to 0FFFFFFF hex

Note
- D1+2 and D1+3 are used only when overwriting data, and only 1) For text and CVS data with no carriage returns (i.e., bits 08 to 11 of C set to 0 hex) or 2) for binary data. Always set D1+2 and D1+3 to 00000000 hex when writing data with carriage returns (i.e., bits 08 to 11 of C set to between 8 and D hex).
- D1 to D1+3 must be in the same data area.

D2: Filename

- D2 is the starting address of the words containing the absolute path and filename in ASCII. Use ASCII a to z, A to Z, and 0 to 9.
- The full path name to the directory containing the data file can be up to 65 characters long, including the starting slash (ASCII 5C). The filename can be up to 8 characters long, but null characters (ASCII 00) are not allowed in the filename because the null character is used to mark the end of the character string. The extension is specified by C, and thus is not included in the file name.
- Separate the directory name and file name with ¥ (#5C).

D2	F1	F2
D2+1	F3	F4
⋮	⋮	⋮
D2+38	F73	F74

Store the character string beginning with the leftmost byte in D2.
The entire pathname and filename can be up to 74 characters (bytes) long, including the initial slash character and ending null character.

S: First Source Word

S specifies the starting address containing the data that will be written to the file memory.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	OK	---	---	---	---
D1, D2									OK	OK	---							
S																		

Flags

Name	Label	Operation
Error Flag	ER	• ON if the file memory type specified in C does not exist. • ON if the settings in C are not within the specified range. • ON if the filename specified in D2 does not satisfy the required conditions. • ON if the File Memory Operation Flag was ON. • ON if a constant was not specified for C (only for CS-series CS1 CPU Units prior to V1). • ON if data specified for D1 is out of range (all CPU Units except for CS-series CS1 CPU Units prior to V1). • ON if an illegal area is specified for S. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.

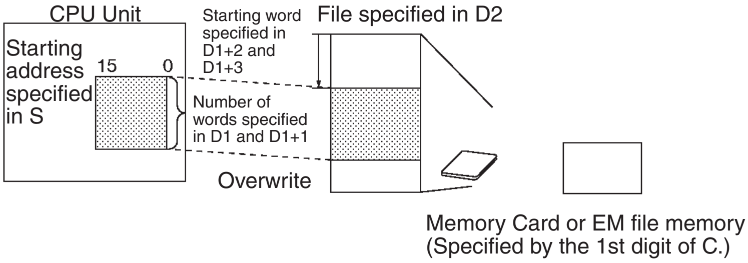
Related Auxiliary Area Flags and Words

Name	Address	Operation
Memory Card Type	A343.00 to A343.02	Contains a binary number indicating the type of Memory Card, if any, that is installed. (0: None, 4: Flash ROM)
Memory Card Format Error Flag	A343.07	ON when the Memory Card is not formatted or a formatting error has occurred.
File Write Error Flag	A343.08	ON when an error occurred when writing to the file.
File Write Impossible Flag	A343.09	ON when the data could not be written because the file was write-protected or there was insufficient free memory.
No File Flag	A343.11	ON when the specified directory does not exist when writing a file.
File Memory Operation Flag	A343.13	ON for any of the following: The CPU Unit has sent a FINS command to itself using CMND(490). FREAD(700) or FWRIT(701) are being executed. The program is being overwritten using a control bit in memory. A simple backup operation is being performed.
Accessing File Flag	A343.14	ON when file data is actually being accessed.
Memory Card Detected Flag	A343.15	ON when a Memory Card has been detected. 0 (OFF) when a Memory Card cannot be detected.
EM File Memory Format Error Flag	A343.06	ON when there is a format error in the starting bank of EM file memory.
Number of Data to Transfer	A346 to A347	The contents of these words indicate the status of data file transfers. When an FWRIT(701) instruction is executed, the number of words or fields to be transferred is written to these words. The value is decremented by 1 as each word is transferred. A346 contains the rightmost 16 bits and A347 contains the leftmost 16 bits of the 32-bit binary value.

Function

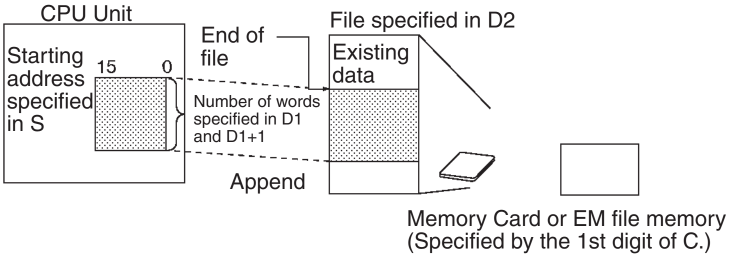
- Overwriting Data in an Existing File (Third Digit of C=1)

  FWRIT(701) uses data area data starting at the word specified in S to overwrite file memory data in the specified data type. It overwrites the number of words or fields specified in D1 and D1+1 in the file specified in D2 (with filename extension .IOM, .TXT, or .CVS) starting at the address specified in D1+2 and D1+3.
  In this case, the data will be overwritten up to the capacity of the existing file. Any data that exceeds the capacity of the existing file will not be written.



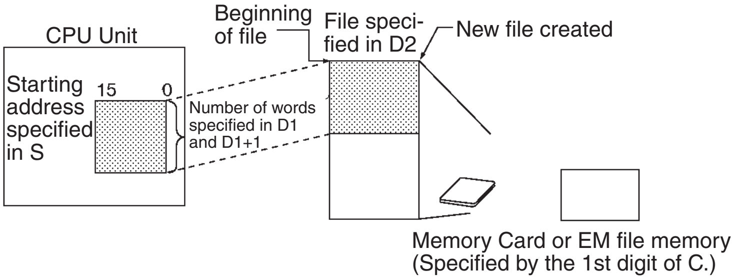
- Appending Data to an Existing File (Third Digit of C=0)

  FWRIT(701) appends data area data starting at the word specified in S to a data file in file memory in the specified data type. It appends the number of words or field specified in D1 and D1+1 to the file specified in D2 (with filename extension .IOM, .TXT, or .CVS).



● Creating a New File with Source Data

If the file specified in D2 does not exist, FWRIT(701) creates a new file with that name and filename extension (.IOM, .TXT, or .CVS) and writes the specified source data in the specified data type starting at the beginning of the file. In this case, it does not matter if appending to overwriting data is specified.



● Data format, carriage return, carriage return position, and number of write fields

- If delimiting is specified, the specified of delimiter is added after every word for word data types and after every two words for double-word data types.
- If non-delimited words or double-words are specified, the data for all fields is written continuously without any delimiters. (The code for a comma is added for comma-delimiting and the code for a tab is added for tab-delimiting.)
- If carriage returns are specified, a carriage return will be added after each set of the specified number of words. If no carriage returns is specified, the data will be written continuously without carriage returns.
- The number of fields specified in the write size is read from I/O memory and written to file memory in the above format.
- If the specified directory does not exist, the File Missing Flag (A34311) will be turned ON and the file data will not be written.

Example: Write the pathname and filename in ASCII beginning with the leftmost byte of D2, as shown in the following example for \ABC\XYZ.IOM. (The extension is added automatically.)

D2	'\'	'A'
D2+1	'B'	'C'
D2+2	'\'	'X'
D2+3	'Y'	'Z'
D2+4	NUL	

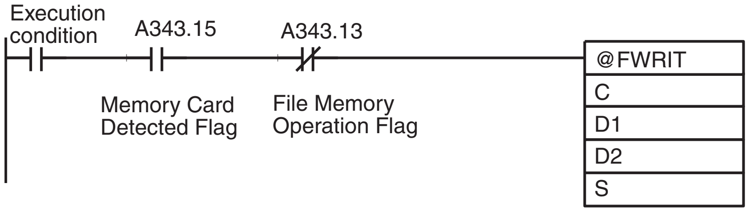
D2	5C	41
→D2+1	42	43
D2+2	5C	58
D2+3	59	5A
→D2+4	00	

Hint

- When another file memory related operation (file memory format, file copy, file delete, etc.) is executed from the ladder program, send the file memory related FINS command to the local CPU Unit with a CMND(490) instruction.

Precaution

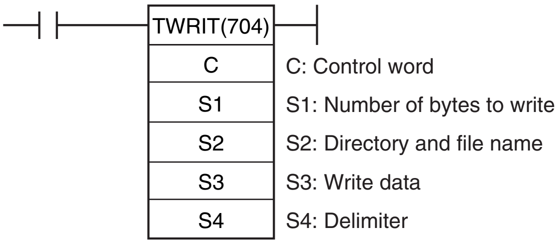
- During normal instruction processing, FWRIT(701) is used only to start writing of the file memory. The instruction execution times given toward the end of this manual are thus the times required to start writing, not to complete it. Actual writing (transfer) is performed by the file access processing in peripheral servicing. Therefore, once FWRIT(701) has been executed, writing is continuously executed even if the execution condition is OFF in following cycles.
  The time required to complete data transfer for FWRIT(701) will depend on the amount of data being transferred, the service time allocated to file access processing, and other conditions. As a guideline, the transfer times for a cycle time of 10 ms for a file in the root directory with the default service time settings will be 1.97 s (new file) or 1.33 s (existing file) for 1,024 words and 6.64 s (new file) or 6.12 s (existing file) for 9,999 words.
- When transfer has been completed, the File Memory Operation Flag (A343.13) will turn OFF. This flag can be used for exclusive control of file memory instructions.
- The source data is read from absolute internal memory addresses in RAM, so the entire block of data will be read even if the data spans two or more data areas. For example, if the first destination address is in the Work Area but the amount of data exceeds the capacity of this area, FWRIT(701) will continue reading data at the beginning of the next area (in this case, the Timer Area). When FWRIT(701) is executed, the number of words or fields specified in D1 and D1+1 is written to A346 and A347 (Number of Data to Transfer) and this value is decremented by 1 as each word or field is transferred. The content of these words can be checked to verify that the expected number of words or fields were transferred.
- The File Memory Operation Flag (A343.13) is turned ON when FWRIT(701) is executed. An error will occur and the instruction will not be executed if A343.13 is already ON.
- If the specified starting word exceeds the number of words in the data file, the File Write Error Flag (A343.08) will be turned ON and the data will not be written.
- The File Write Impossible Flag (A343.09) will be turned ON and the instruction will not be executed if data could not be written because the file was write-protected or there was not enough free memory.
- The File Write Error Flag (A343.08) will be turned ON and the instruction will not be executed if the specified file is not the correct data type or the file data has been corrupted.
- A few seconds is required for the CPU Unit to detect a Memory Card after it has been inserted. If a Memory Card is going to be accessed soon after power is turned ON or after a Memory Card is inserted, use the Memory Card Detected Flag (A343.15) in a NO input condition as shown below to be sure that the Memory Card has been detected.



- The source data words starting at S are accessed and read during the peripheral servicing after FWRIT(701) is executed. If the source data is changed before the file memory write processing is completed, the changed data may be written to the file.

TWRIT

Instruction	Mnemonic	Variations	Function code	Function
WRITE TEXT FILE	TWRIT	@TWRIT	704	Reads ASCII data from I/O memory and stores that data in the Memory Card as a text file (writing a new file or appending a file).

	TWRIT
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
C	Control word	UINT	1
S1	Number of bytes to write	UINT	1
S2	Directory and file name	WORD	1
S3	Write data	WORD	1
S4	Delimiter	WORD	1

C: Control Word

- #0000: Append file.
- #0001: Create new file or overwrite.

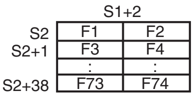
S1: Number of write bytes

Specifies the number of bytes to write in the range 0 to 255 decimal or 0000 to 00FF hexadecimal.

S2: First directory/filename word

- Specifies the first word of the words containing the file's directory path and filename. Input the path and filename in ASCII text.
- Directory name:
  The directory name can be 1 to 65 characters long. If the name is less than 65 characters, do not pad with spaces. Specify the absolute path from the root directory's \ (#5C) character.
- Filename:
  Filename identifier: The identifier can be 1 to 8 characters long. If the name is less than 8 characters, do not pad with spaces. Add a NUL character (#00) at the end of the filename. (The NUL character is not included as one of the 8 characters.)
- Filename extension: None
- Separate the directory name and filename with a \ (#5C) delimiter.

Note The words containing the directory path and filename (starting at S2) must be in the same data area.



Store the character string beginning with the leftmost byte in S2, in the order leftmost byte → rightmost byte and lower word address → higher word address. The directory name and filename can be up to 74 bytes long, including the NULL (00 Hex) at the end of the filename.

S3: First write data word

Specifies the first word (I/O memory data area address) containing the data to be written.

Note It is not necessary for all of the source words (starting at S3) to be in the same data area. The data will be read in PLC memory address order and written as a file.

S4: Delimiter character

Specifies the delimiter characters (up to 2 bytes) for the write data in ASCII. If a delimiter is not required, specify #0000.

Up to 2 bytes can be specified. When 1 byte is being specified, set the rightmost byte to #00.

Typical delimiters (all hexadecimal):

#2C00: Comma (1 byte)
#0A00: Line feed (1 byte)
#0D0A: Carriage return/Line feed (2 bytes)
#0C00: New page (1 byte)
#0900: Tab (1 byte)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
S1																		
S2, S3											---							
S4											OK							

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if there is no Memory Card. • ON if C is not within the specified range of 0000 or 0001. • ON if the filename specified at S2 does not meet the required conditions. • ON if the File Memory Operation Flag is ON. • ON if the data area specified for S3 is an invalid area. • With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. • OFF in all other cases.

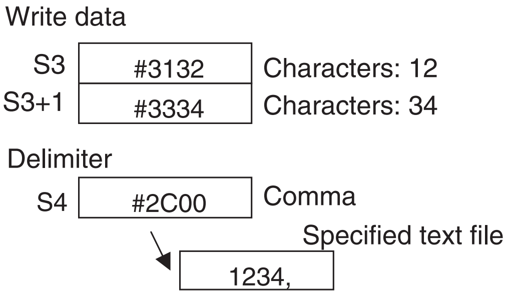
Related Auxiliary Area Flags and Words

Name	Label	Operation
Memory Card Format Error Flag	A343.07	ON when the Memory Card is not formatted or a formatting error has occurred.
File Write Error Flag	A343.08	ON when an error occurred when writing to the file.
File Write Impossible Flag	A343.09	ON when the data could not be written because the file was write-protected or there was insufficient free memory.
No File Flag	A343.11	ON when the specified directory does not exist when writing a file.
File Memory Operation Flag	A343.13	ON for any of the following, otherwise OFF: • The CPU Unit has sent a command to itself using CMND(490). • FREAD(700), FWRIT(701), or TWRIT(704) is being executed. • The program is being overwritten using a control bit in memory. • A simple backup operation is being performed.
Accessing File Flag	A343.14	ON when file data is actually being accessed.
Memory Card Detected Flag	A343.15	ON when a Memory Card has been detected. OFF when a Memory Card could not be detected.
Number of Data Items to Transfer	A346 and A347	The contents of these words indicate the status of data file transfers. When an file write instruction is executed, the number of bytes to be transferred is written to these words. The value is decremented by 1 as each byte is transferred. A346 contains the rightmost 16 bits and A347 contains the leftmost 16 bits of the 32-bit binary value.

Function

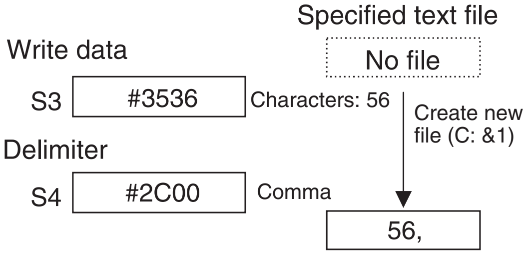
TWRIT(704) writes the number of bytes of data specified in S1, starting from the word specified in S3, to a text file (filename.TXT) in the Memory Card with the filename specified in S2.

A delimiter can be specified in S4 and attached to the end of the text file. The created text file can be referenced later with a text editor.



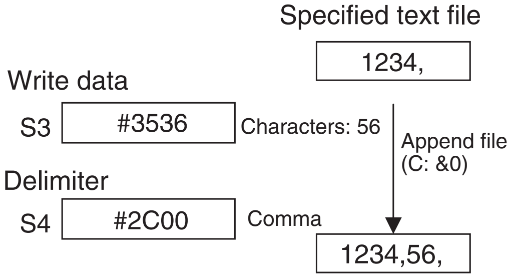
- Creating a New File

  Set C = 0001 and specify a new filename to create a new file.

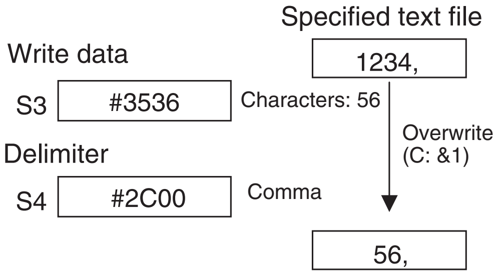


- Appending an Existing File

  Set C = 0000 to append data to an existing file.



  Set C = 0001 and specify an existing filename to overwrite an existing file.



● Data Format

- Store the data in the I/O memory area in order from leftmost byte → rightmost byte and lower word address → higher word address, starting from the leftmost byte of S3.

When Writing the String 12345678

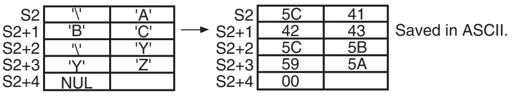
S3	#3132	Characters: 12
S3+1	#3334	Characters: 34
S3+2	#3536	Characters: 56
S3+3	#3738	Characters: 78

● Directory Name and Filename (S2)

- Specify the directory name as the absolute path from the root directory (\). The root directory's \ (#5C) delimiter must be entered. The directory name can be up to 65 characters long. If there are fewer than 65 characters, it is not necessary to add spaces after the directory name. Use \ (#5C) delimiters to separate directory levels. The allowed characters are “a to z”, “A to Z”, and “0 to 9", in ASCII.
- Set the filename as 1 to 8 ASCII characters, using only the “a to z”, “A to Z”, and “0 to 9" characters. If there are fewer than 8 characters, it is not necessary to add spaces after the filename. Always insert an NULL (#00) character after the filename.
- The filename extension is fixed to “.TXT”, so it is not specified.

- Store the directory name and filename in ASCII and in order from leftmost byte → rightmost byte and lower word address → higher word address, starting from the leftmost byte of S2.
- If the specified directory does not exist, the No File Flag (A343.11) will be turned ON and the file will not be overwritten.

Example: Example: Writing to Directory \ABC and Filename XYZ



Hint

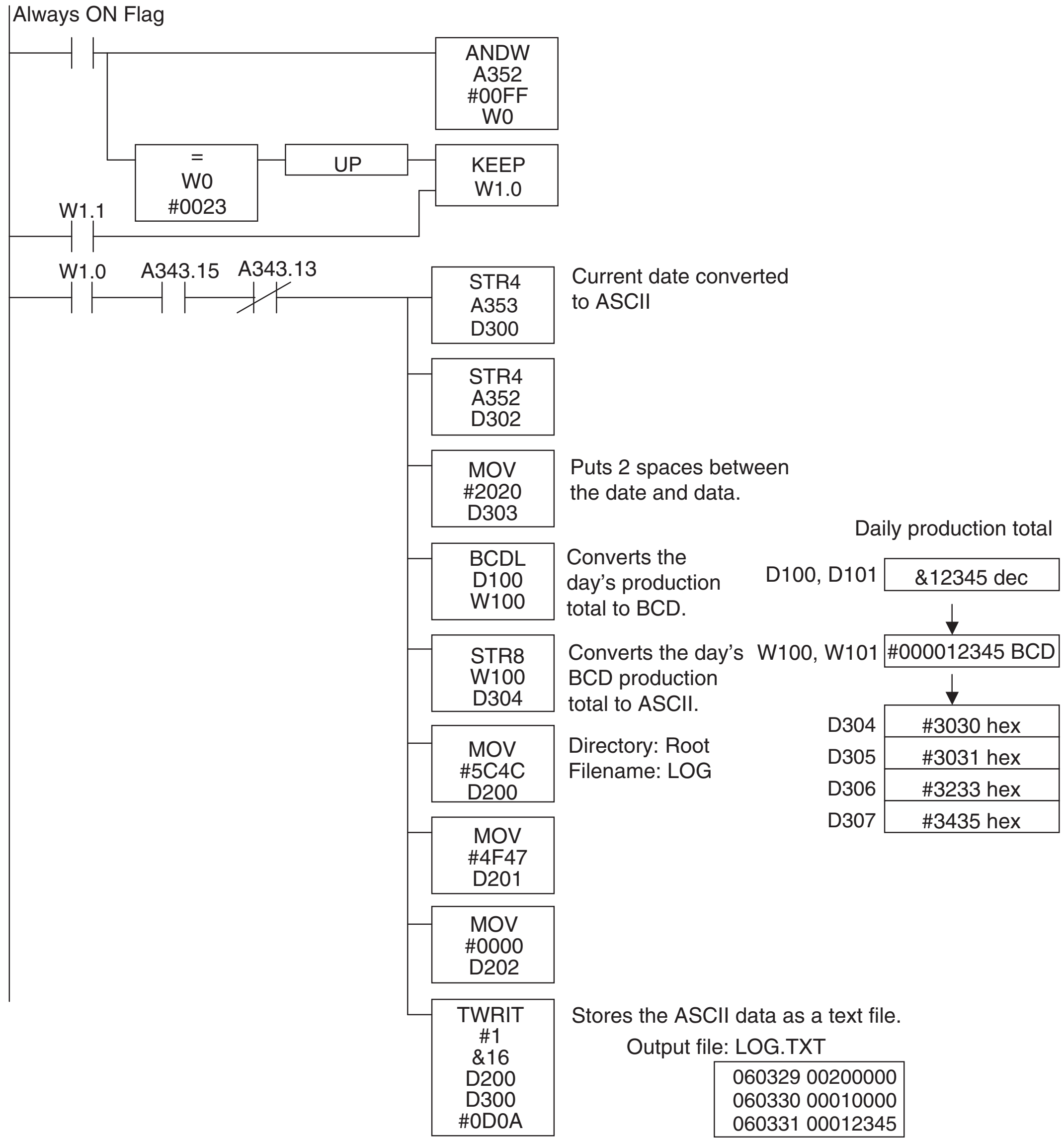
- When another file memory related operation (file memory format, file copy, file delete, etc.) is executed from the ladder program, send the file memory related FINS command to the local CPU Unit with a CMND(490) instruction.

Precaution

- During normal instruction execution processing, TWRIT(704) is used only to start the writing of the file memory. The instruction execution times given toward the end of this manual are thus the times required to start writing, not to complete it.
  Actual writing (transfer) is performed by the file access processing in peripheral servicing. Therefore, once TWRIT(704) has been executed, writing is continuously executed even if the execution condition is OFF in following cycles.
  The time required to complete data transfer for TWRIT(704) will depend on the amount of data being transferred, the service time allocated to file access processing, and other conditions. As a guideline, if the cycle time is 10 ms and the file is in the root directory, it will take about 440 ms (new file) or 260 ms (existing file) to write 100 bytes, and about 450 ms (new file) or 270 ms s (existing file) to write 255 bytes. These guideline values will vary widely depending on the type of Memory Card being used and the number of files in the Memory Card.
- When transfer has been completed, the File Memory Operation Flag (A343.13) will turn OFF. This flag can be used for exclusive control of file memory instructions.
- The source data is read from absolute PLC memory addresses in RAM, so the entire block of data will be read even if the data spans two or more data areas. For example, if the first source address is in the Work Area but the amount of data exceeds the capacity of this area, TWRIT(704) will continue reading data at the beginning of the next area (in this case, the Timer Area).
- When TWRIT(704) is executed, the “number of write bytes” specified in S1 is written to A346 and A347 (Number of Data Items to Transfer) and this value is decremented by 1 as each byte is transferred. The content of these words can be checked to verify that the expected number of bytes were transferred.
- The File Memory Operation Flag (A343.13) is turned ON when TWRIT(704) is executed. An error will occur and the instruction will not be executed if A343.13 is already ON.
- The File Write Impossible Flag (A343.09) will be turned ON and the instruction will not be executed if data could not be written because the file was write-protected or there was not enough free memory.
- A few seconds is required for the CPU Unit to detect a Memory Card after it has been inserted. If a Memory Card is going to be accessed soon after power is turned ON or after a Memory Card is inserted, use the Memory Card Detected Flag (A343.15) in a NO input condition as shown in the example below to be sure that the Memory Card has been detected.

Example Programming

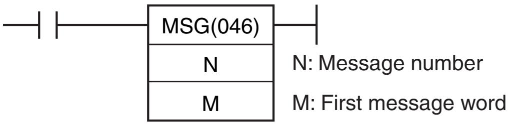
This example records the daily production total (number of units produced) in D100 and D101 in 8-digit hexadecimal. Every day at 23:00, the program converts the daily production total to BCD format and appends the file LOG.TXT in the Memory Card's root directory.



Display Instructions

MSG

Instruction	Mnemonic	Variations	Function code	Function
Display Instructions	MSG	@MSG	046	Reads the specified sixteen words of extended ASCII and displays the message on a Peripheral Device such as a Programming Console.

	MSG
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Message number	UINT	1
M	First message word	UINT	Variable

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
M												---						

Flags

Name	Label	Operation
Error Flag	ER	• ON if the content of S is not 0000 to 0007 hexadecimal. • OFF in all other cases.

Function

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, a Programming Console 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.

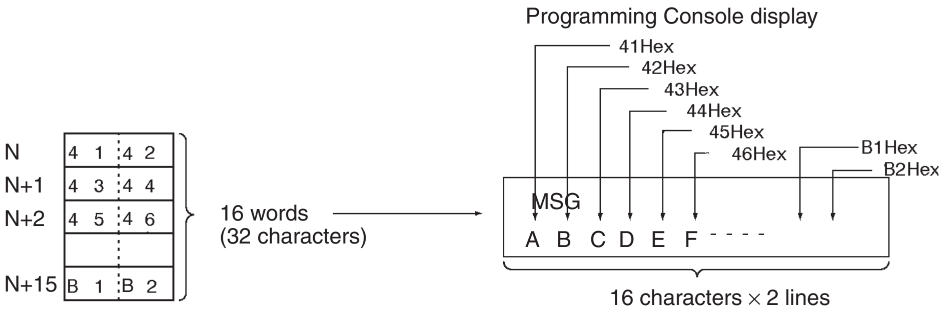
To clear a message that has been registered, execute MSG(046) with S set to the message number of the message you want to clear and N set to a constant (0000 to FFFF).

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.

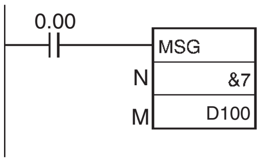
Precautions

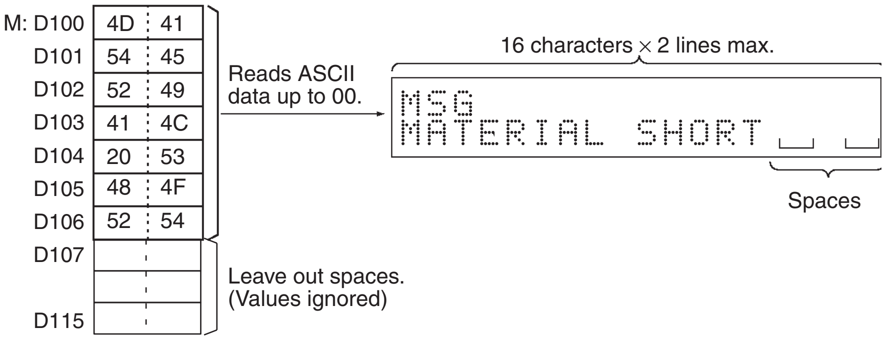
- Registered messages are updated each time MSG(046) is executed.
- All message characters after the null character (00) are converted to spaces in the Programming Console display.
- The character stored in the leftmost byte is displayed before the character in the rightmost byte.

Example Programming



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 Peripheral device.





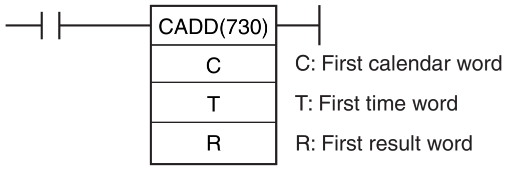
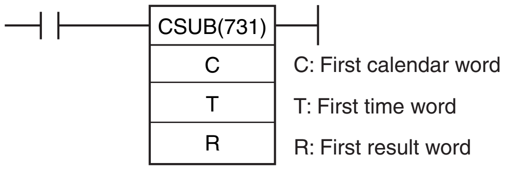
● ASCII

		Four leftmost bits															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Four rightmost bits	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	゜		

Clock Instructions

CADD/CSUB

Instruction	Mnemonic	Variations	Function code	Function
CALENDAR ADD	CADD	@CADD	730	Adds time to the calendar data in the specified words.
CALENDAR SUBTRACT	CSUB	@CSUB	731	Subtracts time from the calendar data in the specified words.

	CADD	CSUB
Symbol		

Applicable Program Areas

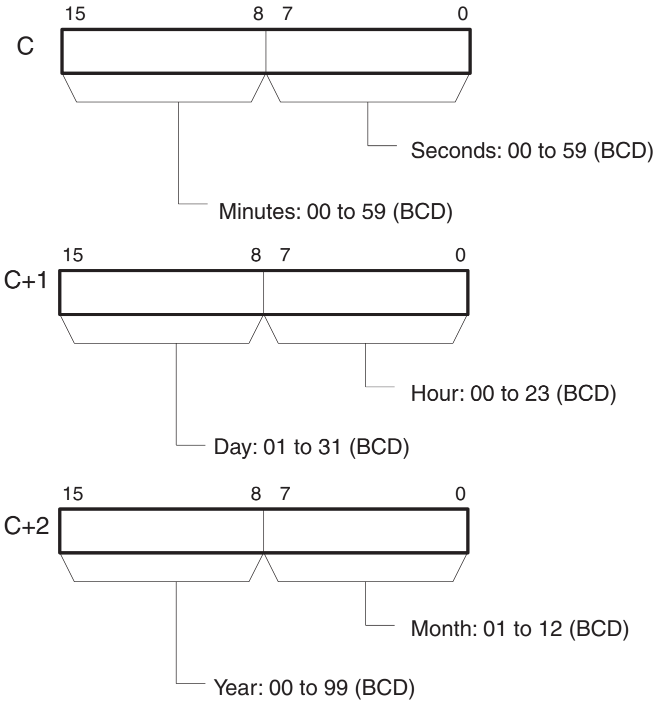
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

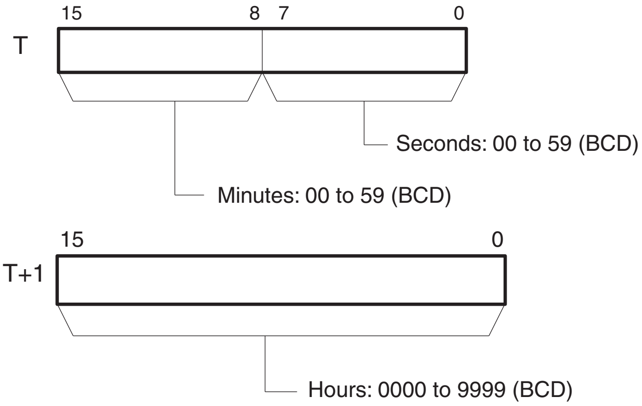
Operand	Description	Data type	Size
C	First calendar word	WORD	3
T	First time word	DWORD	2
R	First result word	WORD	3

- **CADD**

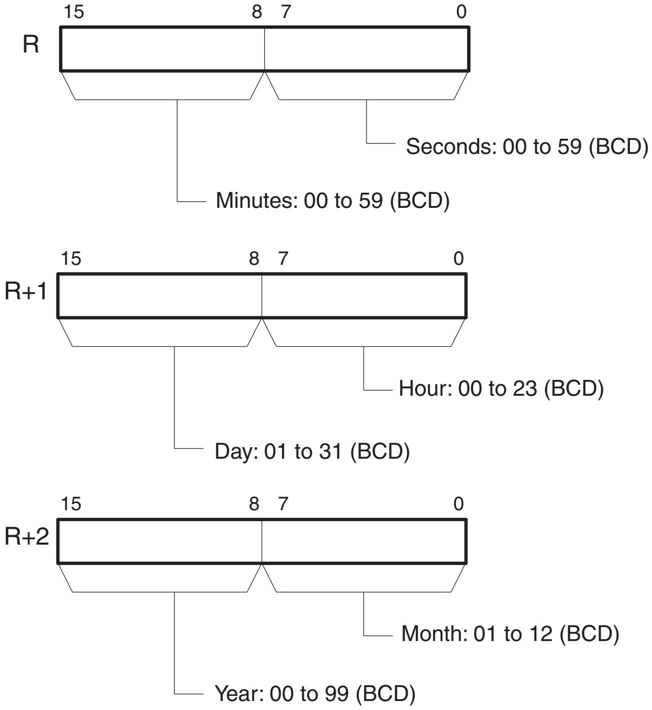
C through C+2: Calendar Data



T and T+1: Time Data

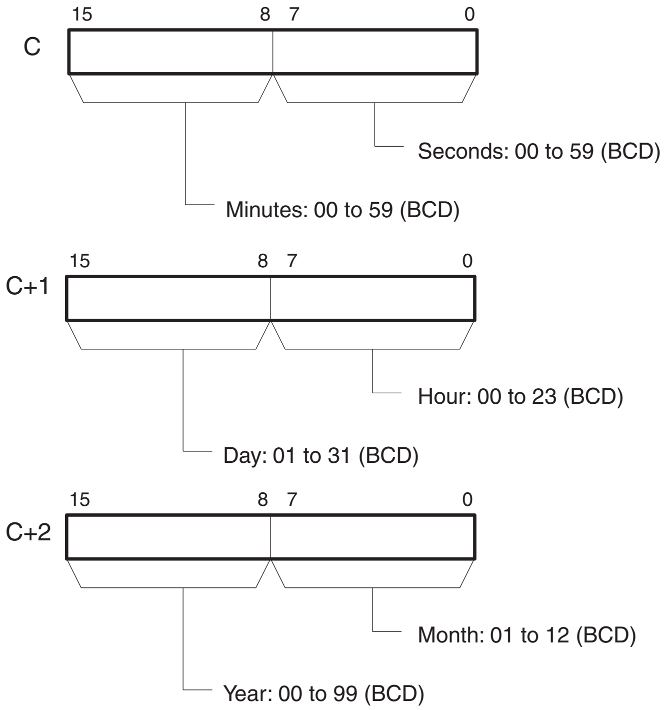


R through R+2: Result Data

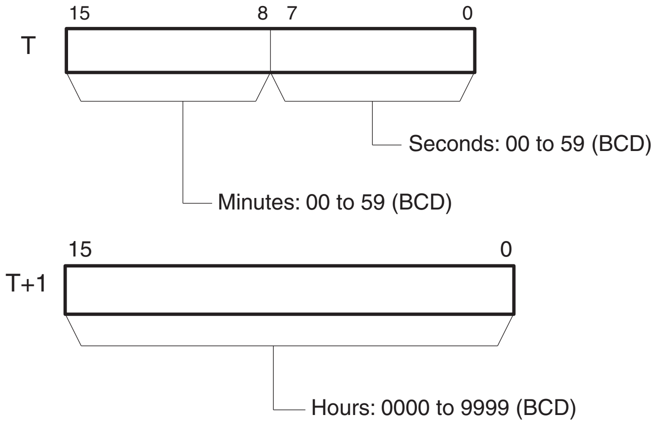


CSUB

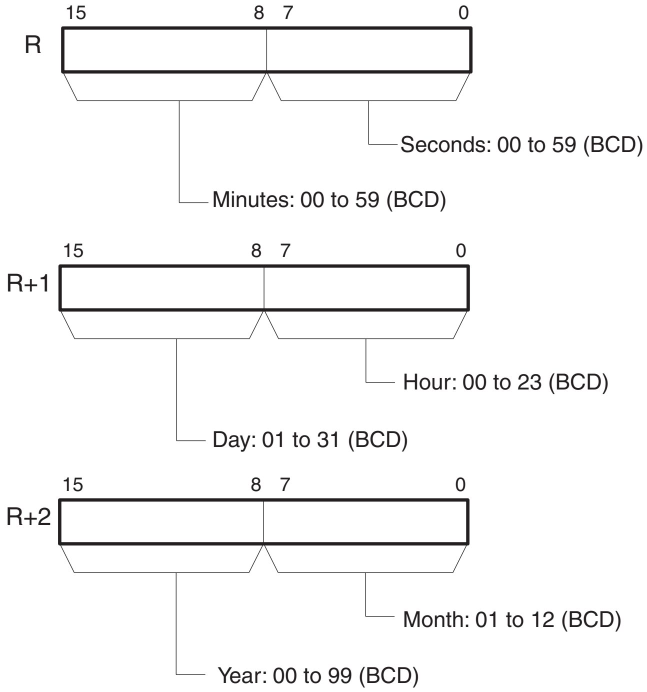
C through C+2: Calendar Data



T and T+1: Time Data



R through R+2: Result Data



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con- stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
T											OK							
R											---							

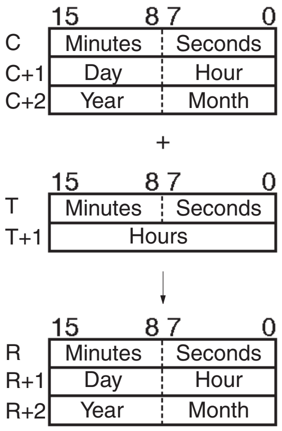
Flags

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.
Equal Flag	=	• ON when the result of a CSUB instruction is 0 • OFF in all other cases.

Function

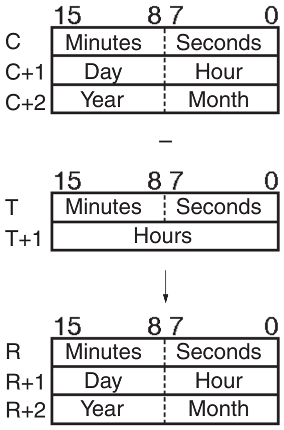
● CADD

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.



● CSUB

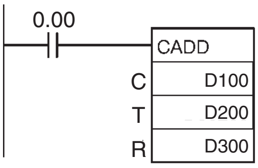
CSUB(731) subtracts the time data (words T and T+1) from the calendar data (words C through C+2) to and outputs the resulting calendar data to R through R+2.

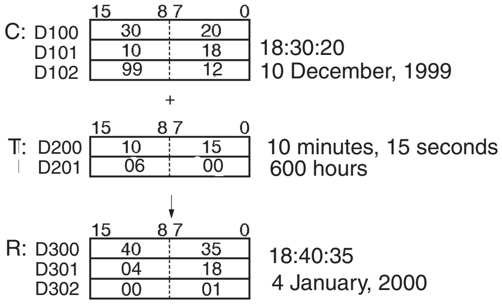


Example Programming

● CADD

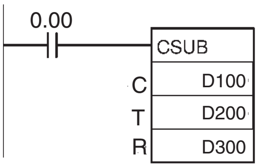
When CIO 0.00 turns ON in the following example, the calendar data in D100 through D102 (year, month, day, hour, minutes, seconds) is added to the time data in D200 and D201 (hours, minutes, seconds) and the result is output to D300 through D302.

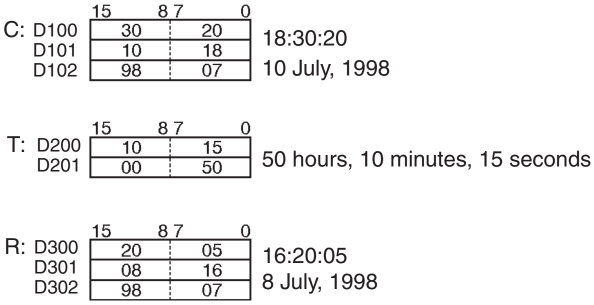




● CSUB

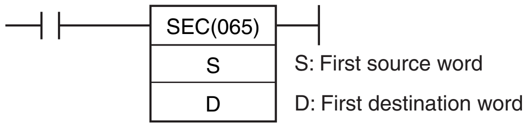
When CIO 0.00 turns ON in the following example, the time data in D200 and D201 (hours, minutes, seconds) is subtracted from the calendar data in D100 through D102 (year, month, day, hour, minutes, seconds) and the result is output to D300 through D302.





SEC

Instruction	Mnemonic	Variations	Function code	Function
HOURS TO SECONDS	SEC	@SEC	065	Converts time data in hours/minutes/seconds format to an equivalent time in seconds only.

	SEC
Symbol	

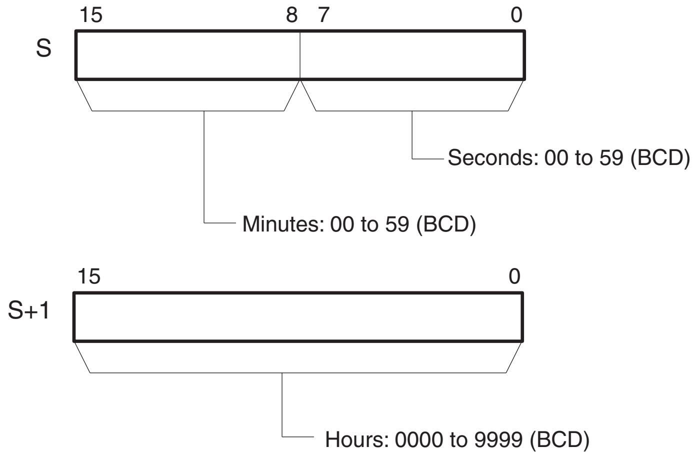
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

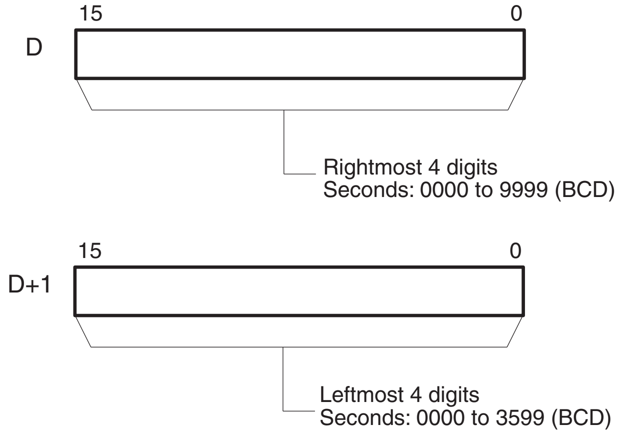
Operands

Operand	Description	Data type	Size
S	First source word	DWORD	2
D	First destination word	DWORD	2

S and S+1: Source Data (hours/minutes/seconds)



D and D+1: Result Data (seconds only)



Note S, S+1, D and D+1 must be in the same data area.

- **Operand Specifications**

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
D											---							

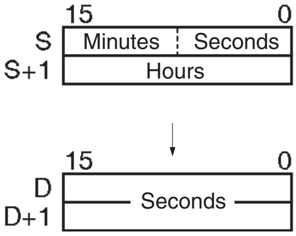
Flags

Name	Label	Operation
Error Flag	ER	• ON when S+1 (time data) is not BCD data • 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.
Equal Flag	=	• ON if the content of D is 0000 after the operation. • OFF in all other cases.

Function

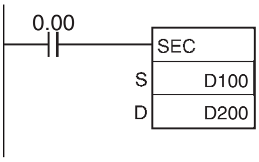
SEC(065) converts the 8-digit BCD hours/minutes/seconds data in S and S+1 to 8-digit BCD seconds-only data and outputs the result to D and D+1.

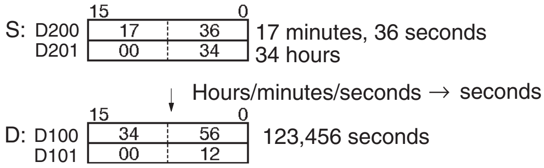
The maximum value for the source data is 9,999 hours, 59 minutes, and 59 seconds (35,999,999 seconds).



Example Programming

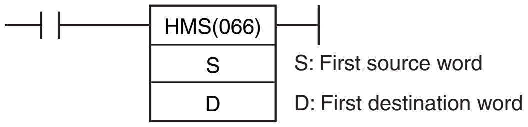
When CIO 0.00 turns ON in the following example, the hours/minutes/seconds data in D200 and D201 (34 hours, 17 minutes, and 36 seconds) is converted to seconds-only data and the result is output to D100 and D101.





HMS

Instruction	Mnemonic	Variations	Function code	Function
SECONDS TO HOURS	HMS	@HMS	066	Converts seconds data to an equivalent time in hours/minutes/seconds format.

Symbol	HMS
	

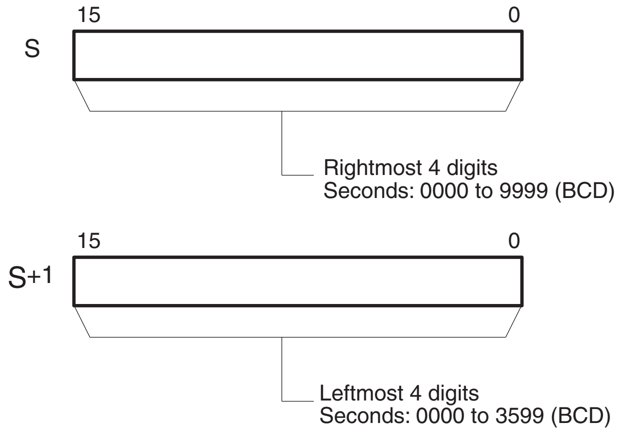
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

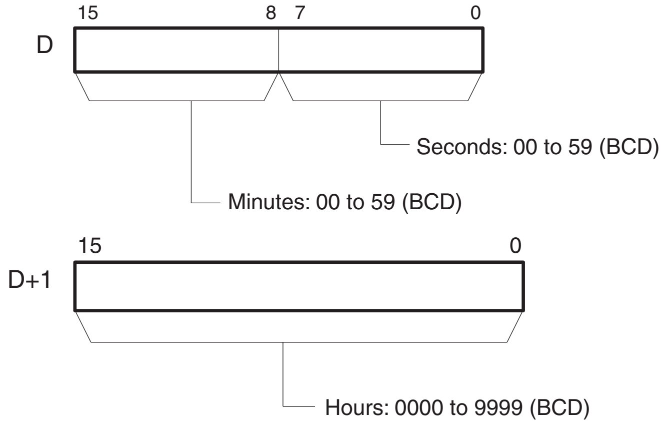
Operands

Operand	Description	Data type	Size
S	First source word	DWORD	2
D	First destination word	DWORD	2

S and S+1: Source Data (seconds)



D and D+1: Result Data (hours/minutes/seconds)



Note S, S+1, D and D+1 must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	OK	---	---	---	---
D											---							

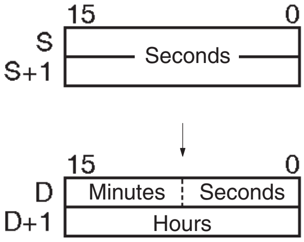
Flags

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.
Equal Flag	=	• ON if the content of D is 0000 after the operation. • OFF in all other cases.

Function

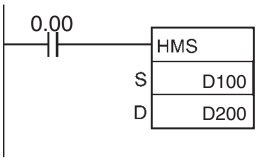
HMS(066) converts the 8-digit BCD seconds-only data in S and S+1 to 8-digit BCD hours/minutes/seconds data and outputs the result to D and D+1.

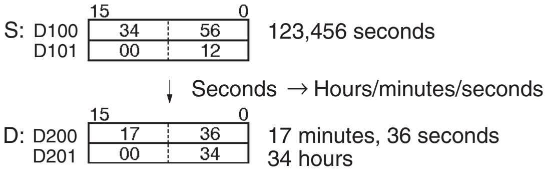
The maximum value for the source data is 35,999,999 seconds (9,999 hours, 59 minutes, and 59 seconds).



Example Programming

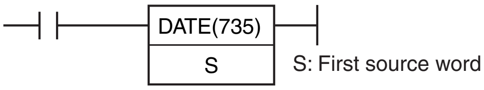
When CIO 0.00 turns ON in the following example, the seconds data in D100 and D101 (123,456 seconds) is converted to hours/minutes/seconds data and the result is output to D200 and D201.





DATE

Instruction	Mnemonic	Variations	Function code	Function
CLOCK ADJUSTMENT	DATE	@DATE	735	Changes the internal clock setting to the setting in the specified source words.

Symbol	DATE
	

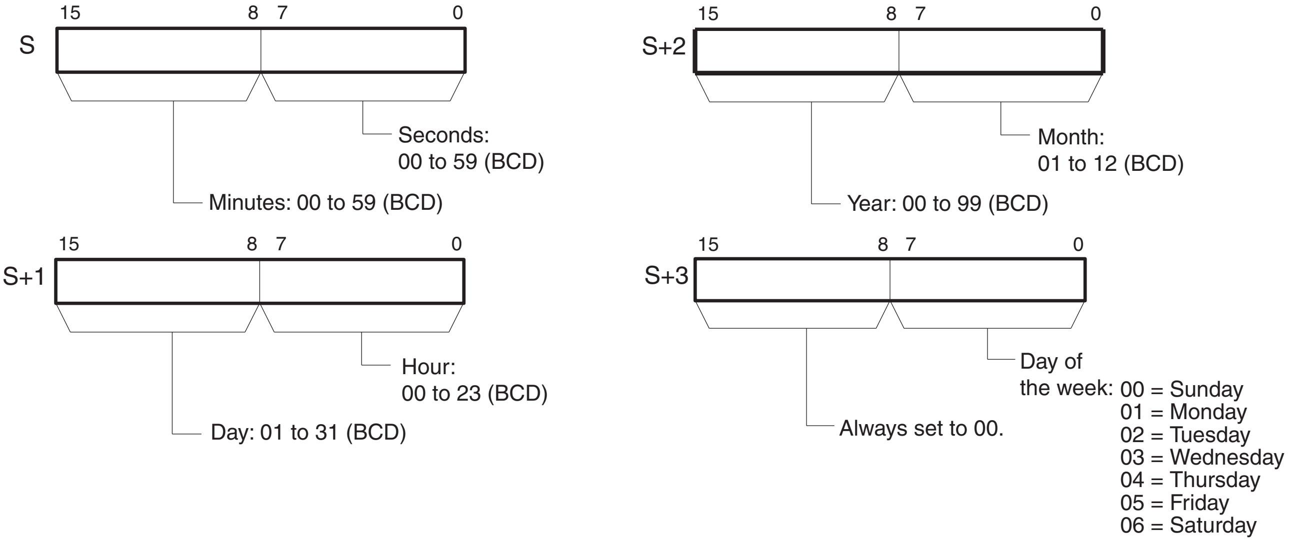
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	LWORD	4

S through S+3: New Clock Setting



Note S through S+3 must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

Flags

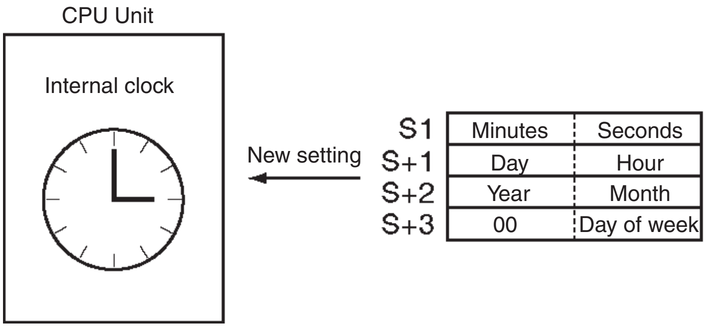
Name	Label	Operation
Error Flag	ER	• ON if the new clock setting in S through S+3 is not within the specified range. • OFF in all other cases.

Related Auxiliary Area Words and Bits

Name	Address	Operation
Clock data	A351 to A354	A351.00 to A351.07: Seconds (00 to 59) (BCD) A351.08 to A351.15: Minutes (00 to 59) (BCD) A352.00 to A352.07: Hours (00 to 23) (BCD) A352.08 to A352.15: Day of the 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 the week (00 to 06) (BCD) 00: Sunday, 01: Monday, 02: Tuesday, 03: Wednesday, 04: Thursday, 05: Friday, 06: Saturday

Function

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).



S1	Minutes	Seconds
S+1	Day	Hour
S+2	Year	Month
S+3	00	Day of week

Hint

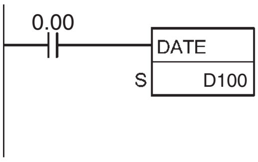
The internal clock setting can also be changed from a Peripheral Device or the CLOCK WRITE FINS command (0702).

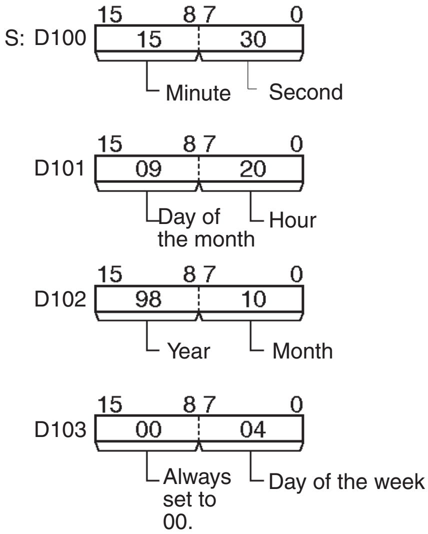
Precaution

An error will not be generated even if the internal clock is set to a non-existent date (such as November 31).

Example Programming

When CIO 0.00 turns ON in the following example, the internal clock is set to 20:15:30 on Thursday, October 9, 1998.





Debugging Instructions

TRSM

Instruction	Mnemonic	Variations	Function code	Function
Trace Memory Sampling	TRSM	---	045	When TRSM(045) is executed, the status of a pre-selected bit or word is sampled and stored in Trace Memory.

Symbol	TRSM
	

Applicable Program Areas

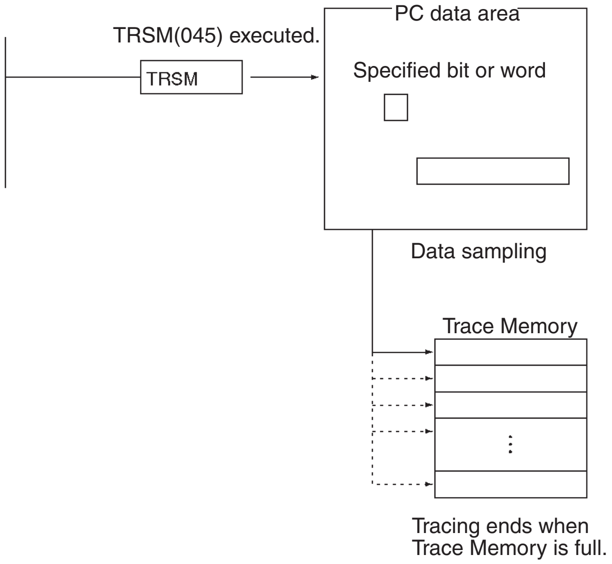
Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

There are no flags affected by this instruction.

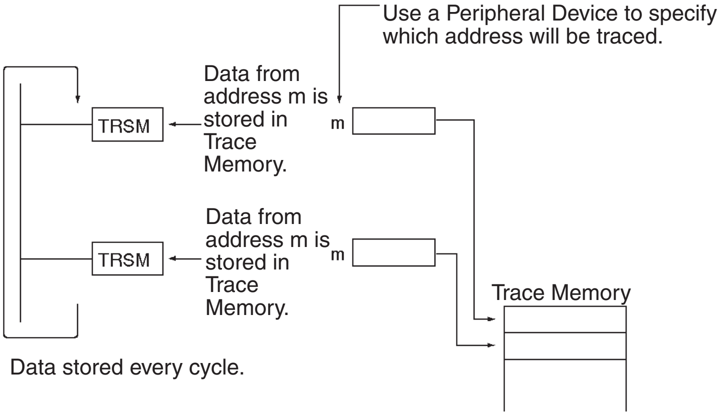
Function

Before TRSM(045) is executed, the bit or word to be traced must be specified with a Peripheral Device. 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 a Peripheral Device when necessary.



- This instruction only indicates when the specified data will be sampled. All other settings and data trace operations are set with a Peripheral Device.
- 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.



Refer to the Peripheral Device's Operation Manual for details on data tracing.

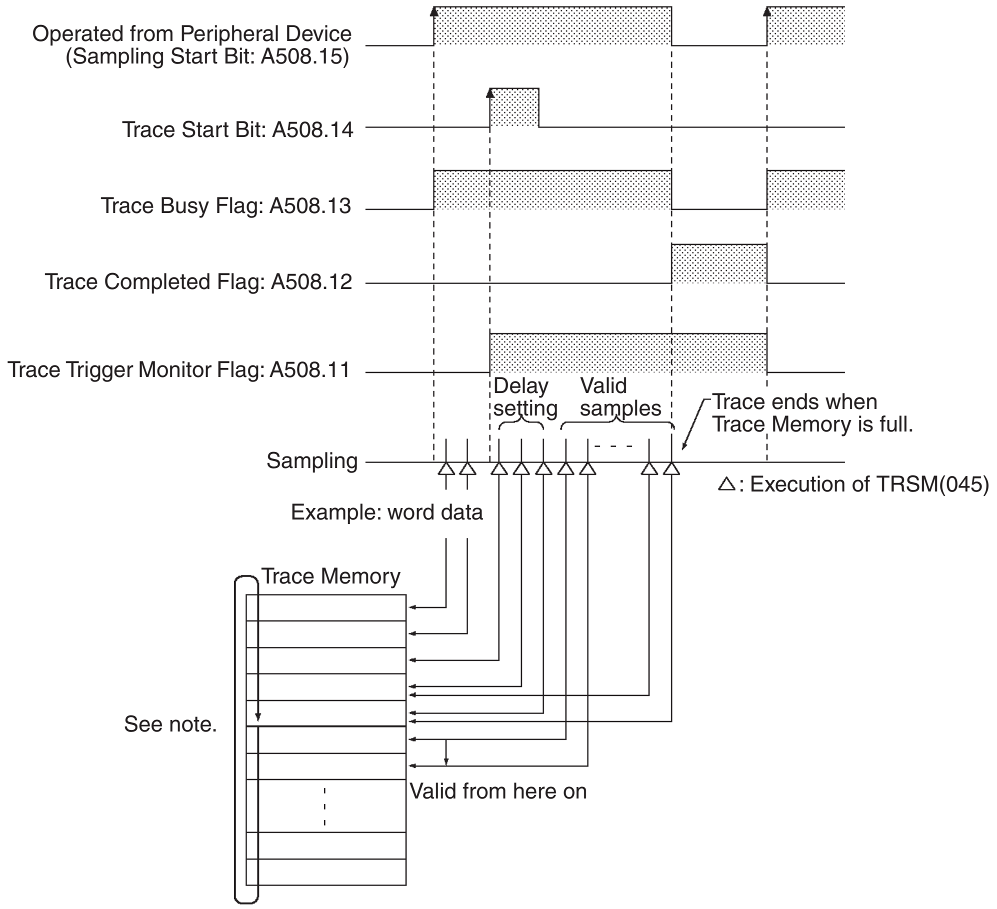
Hint

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).

The data-tracing operations performed with the Peripheral Device are summarized in the following list.

1. Set the following parameters with the Peripheral Device.
    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 Peripheral Device, 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 Peripheral Device.

● Timing Chart



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.

Precaution

TRSM(045) is processed as NOP(000) when data tracing is not being performed or when the sampling interval set in the parameters of data tracing is not set to sample on TRSM(045) instruction execution.

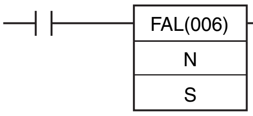
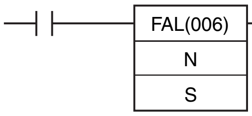
Do not turn the Sampling Start Bit (A508.15) ON or OFF from the program. This bit must be turned ON and OFF from a Peripheral Device.

When a TRSM instruction is used between JMP and JME and the JMP condition is OFF, the TRSM instruction will not be executed.

Failure Diagnosis Instructions

FAL

Instruction	Mnemonic	Variations	Function code	Function
FAILURE ALARM	FAL	@FAL	006	Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop PLC operation.

Symbol	FAL	
	Generating or Clearing User-defined Non-fatal Errors	Generating Non-fatal System Errors
	 N: FAL number S: First message word or constant (0000 to FFFF)	 N: FAL number (value in A529) S: First word containing the error code and error details

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	FAL number	Constants only	1
S	First message word or constant / First word containing the error code and error details	WORD	Variable

● Generating or Clearing User-defined Non-fatal Errors

Note The value of operand N must be different from the content of A529 (the system-generated FAL/FALS number).

	Generates a non-fatal error	Clears all non-fatal errors
N	1 to 511 (These FAL numbers are shared with FALS numbers.)	0
S	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 Programming Device. #0000 to #FFFF: Generates a non-fatal error with the corresponding FAL number (no message).	#FFFF: Clears all non-fatal errors. #0001 to #01FF: Clears the non-fatal error with the corresponding FAL number. Other: Clears the most serious non-fatal error.

● Generating Non-fatal System Errors

Note The value of operand N must be the same as the content of A529 (the system-generated FAL/FALS number).

	Generates a non-fatal error
N	1 to 511 (These FAL numbers are shared with FALS numbers.)
S	Error code that will be generated. (See Function below.)
S+1	Error details code that will be generated. (See Function below.)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK				OK				

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.

Related Auxiliary Area Words and Bits

● 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 numbers1 to 511 decimal.

● 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.

Clearing Non-fatal Errors without a Programming Device

● 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.

● 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.

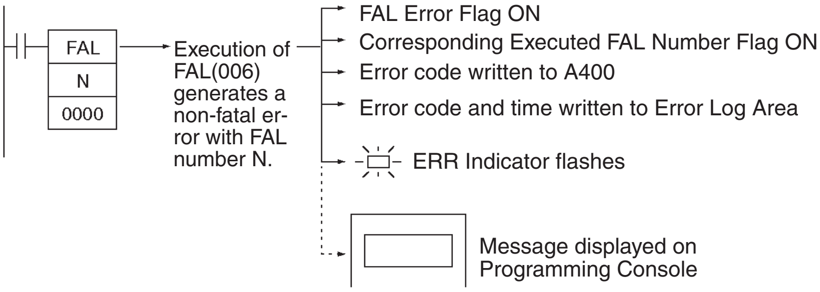
Function

● Generating Non-fatal User-defined Errors

The following table shows the error codes and FAL Error Flags for FAL(006).

FAL number	1 to 511 decimal
FAL error codes	4101 to 42FF
Executed FAL Number Flags	A360.01 to A391.15

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. 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).
4. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).

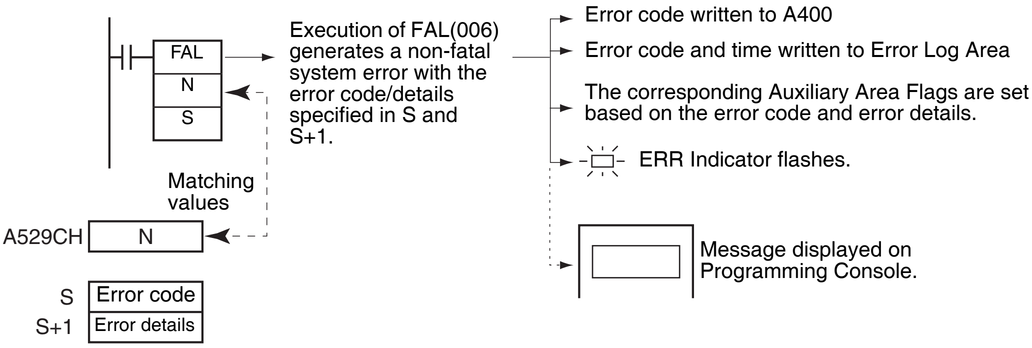
Note The error record will not be written to the Error Log Area if the *Don't register FAL to error log* Option in the PLC Setup is selected.

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).

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.

● 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. 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 Programming Console.

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.
Refer to the *CS/CJ Series Operation Manual* or the *CJ2 CPU Unit Hardware Operation Manual* (W472).

The following table shows how to specify error codes and error details in S and S+1.

Error name	S	S+1
Interrupt Task Error	008B hex	• Bit 15 OFF: Interrupt task error Bits 00 to 14: Task number of interrupt task where error occurred. • 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
Basic I/O Error	009A hex	Rack location of Unit where error occurred • Bits 08 to 15: Rack number (binary) of Rack where the affected Unit is mounted • Bits 00 to 07: Slot number (binary) of slot where the affected Unit is mounted
PLC Setup Error	009B hex	PLC Setup Error Location 0000 to FFFF hex
I/O Table Verification Error	00E7 hex	--- (not fixed)
Non-fatal Inner Board Error	02F0 hex	Inner Board Error Information • Bits 00 to 03: Invalid • Bits 04 to 15: Error defined by the Inner Board
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)
SYSMAC BUS Error	00A0 hex	SYSMAC BUS Master Unit's unit number: 0000 or 0001 hex
Battery Error	00F7 hex	--- (not fixed)
CPU Bus Unit Setup Error	0400 hex	CPU Bus Unit's unit number: 0000 to 000F hex
Special I/O Unit Setup Error	0500 hex	Special I/O Unit's unit number:0000 to 005F hex

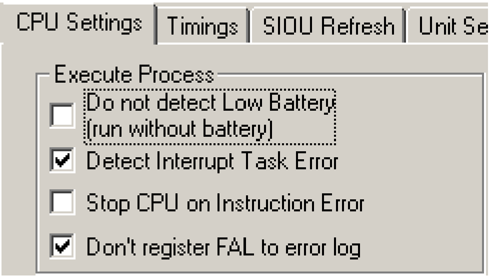
Disabling Error Log Entries of User-defined Errors

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.

Note Even though the error will not be recorded in the Error Log, the FAL Error Flag (402.15) 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 screen capture shows the PLC Setup setting from the CX-Programmer.



Execute Process

☐ Do not detect Low Battery (run without battery)

☑ Detect Interrupt Task Error

☐ Stop CPU on Instruction Error

☑ Don't register FAL to error log

- The following table shows the PLC Setup setting from the Programming Console.

Programming Console setting address	Word	129
	Bit	15
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)	

Note 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)

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 Peripheral 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 a Programming Console 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.

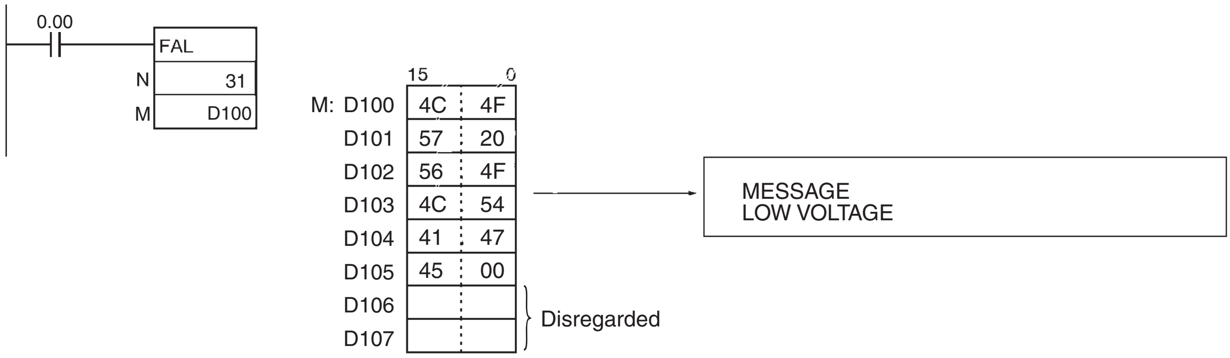
Example Programming

● 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. The FAL Error Flag (A402.15) will be turned ON.
2. The corresponding Executed FAL Number Flag (A361.15) will be turned ON.
3. The corresponding error code (411F) will be written to 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 Peripheral Device.

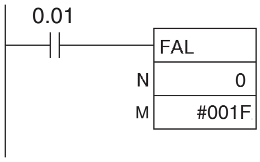
Note If a message is not required, specify a constant for S.



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.

● 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 (A361.15), and turn OFF the FAL Error Flag (A402.15).

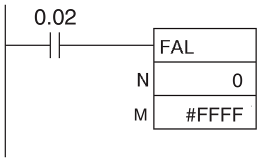


Set N to 0 to clear errors.

Set M to the desired FAL number (031(001F)).

● 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).

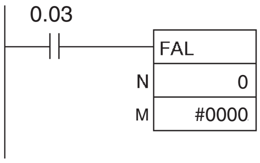


Set N to 0 to clear errors.

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.



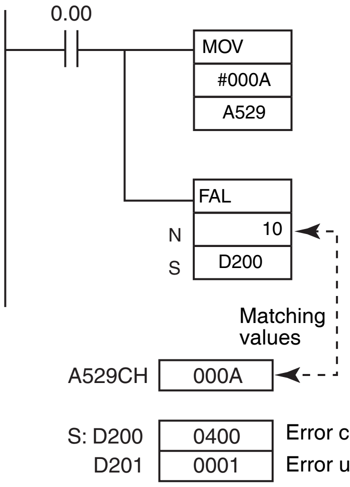
Set N to 0 to clear errors.

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. The specified error code (0400) 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 Setup Error Flag (A402.03) and CPU Bus Unit Setup Error Flag for unit number 1 (A427.01) 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 Console indicating that an error has occurred with CPU Bus Unit 1.

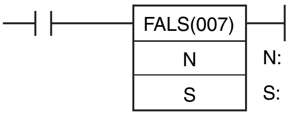
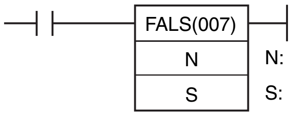


Error code: 0400 (CPU Bus Unit Setup Error)

Error unit number: 1

FALS

Instruction	Mnemonic	Variations	Function code	Function
SEVERE FAILURE ALARM	FALS	---	007	Generates user-defined fatal errors. Fatal errors stop PLC operation.

Symbol	FALS	
	Generating or Clearing User-defined Fatal Errors	Generating Fatal Errors from the System
	 N: FAL number S: First message word or constant (0000 to FFFF)	 N: FAL number (value in A529) S: First word containing the error code and error details

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	FAL number	Constants only	1
S	First message word or constant / First word containing the error code and error details	WORD	Variable

● Generating User-defined Fatal Errors

Note The value of operand N must be different from the content of A529 (the system-generated FAL/FALS number).

	Generates a non-fatal error
N	1 to 511 (These FALS numbers are shared with FALS numbers.)
S	Specifies the first of eight words containing an ASCII message to be displayed on the Programming Device. 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).

	Generates a non-fatal error
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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK				OK				

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.

Related Auxiliary Area Words and Bits

● 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

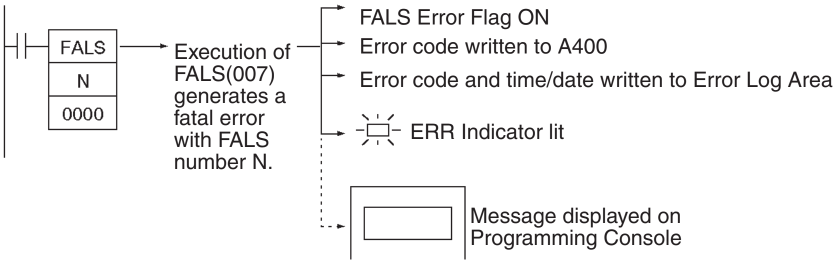
Name	Address	Operation
Error Log Area	A100 to A199	The Error Log Area contains the error codes and time/date of occurrence for the most recent 20 errors, including errors generated by FALS(007).
Error code	A400	When an error occurs its error code is stored in A400. The error codes for FALS numbers 0001 to 01FF (1 to 511 decimal) are C101 to C2FF, respectively. **Note** If two or more errors occur simultaneously, the error code of the most serious error will be stored in A400.

Function

● Generating Fatal User-defined Errors

FALS number	1 to 511
FALS error codes	C101 TO C2FF

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. 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 Peripheral Device).

Note 1 If an error that is more serious (including fatal system errors) than an error registered with this instruction occurs simultaneously, the error code of that error will be set in error code A400.

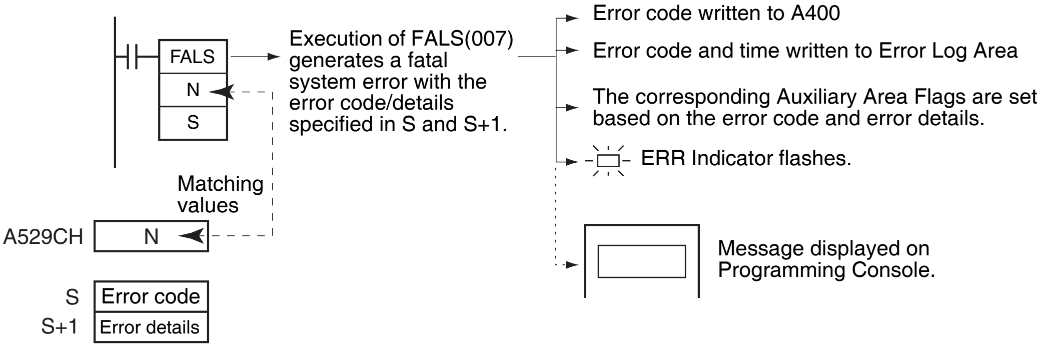
2 The end code for the message is the null character (00 hexadecimal). All 16 characters in words S to S+7 will be displayed if the null character is omitted.

3 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.

4 When a user-defined fatal error is registered, the I/O memory and output status from output units will be as indicated below.

		I/O memory	Output status from output units
IOM Hold Bit (A500.12).	ON	Hold	OFF
	OFF	Hold	OFF

● 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. 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 Console.

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 *CS Series or CJ Series Operation Manual* or the *CJ2 CPU Unit Hardware Operation Manual* (W472) 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).

		Status of I/O memory	Status of outputs on Output Units
IOM Hold Bit (A500.12)	ON	Retained	OFF
	OFF	Cleared	OFF

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 05: Registered I/O table Bit 07: Routing table Bit 08: CPU Bus Unit Setup Bit 09: Memory Card transfer error Bits 10 to 15: Invalid
I/O Bus Error	80C0 hex	Bits 00 to 07: Slot number where the I/O Bus error occurred Slot 0 to 9: 00 to 09 hex Slot unknown: 0F hex Bits 08 to 15: Rack number where the I/O Bus error occurred Slot 0 to 7: 00 to 07 hex Rack unknown: 0F hex
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
Rack Number Duplication Error	80EA hex	Duplicated Rack number (overlapping word allocations) 0000 to 0006 hex
Fatal Inner Board Error	82F0 hex	Error Cause Bits 00 to 03: Error defined by Inner Board Bits 04 to 15: Invalid
Too Many I/O Points Error	80E1 hex	Bits 13 to 15: Error Cause Bits 00 to 12: Details • Total number of I/O points is too high. Bits 13 to 15: 000 Bits 00 to 12: Number of I/O points (binary) • Number of interrupt inputs is too high. Bits 13 to 15: 001 Bits 00 to 12: Number of interrupt inputs (binary) Bits 00 to 12: All zeroes • A Slave Unit’s unit number is duplicated or a C500 Slave Unit has more than 320 I/O points. Bits 13 to 15: 010 Bits 00 to 12: Slave Unit’s unit number (binary) • The unit number of an I/O Interface (excluding Slave Racks) is duplicated. Bits 13 to 15: 011 Bits 00 to 12: Unit number (binary) • A Master Unit’s unit number is duplicated or outside of the allowed setting range. Bits 13 to 15: 100 Bits 00 to 12: Master Unit’s unit number (binary) • Too many Pulse I/O Modules are connected (CJ2M only). Bits 13 to 15: 100 Bits 00 to 12: 000 • The number of Expansion Racks is too high. Bits 13 to 15: 101 Bits 00 to 12: Number of Expansion Racks (binary) • C200H Special I/O Unit or Remote I/O was not recognized. Bits 13 to 15: 110
I/O Table Setting Error	80E0 hex	0000 hex
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/EM BCD error Bit 08: Instruction error Bits 00 to 07: Invalid
Cycle Time Overrun Error	809F hex	0000 hex

● 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 a Programming Console 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.

● 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 a Programming Device or turn the PLC OFF and then ON again.

Precaution

Unlike user-defined fatal errors, system errors generated by FALS(007) will clear I/O memory if the IOM Hold Bit is OFF.

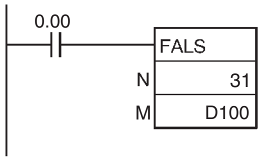
Example Programming

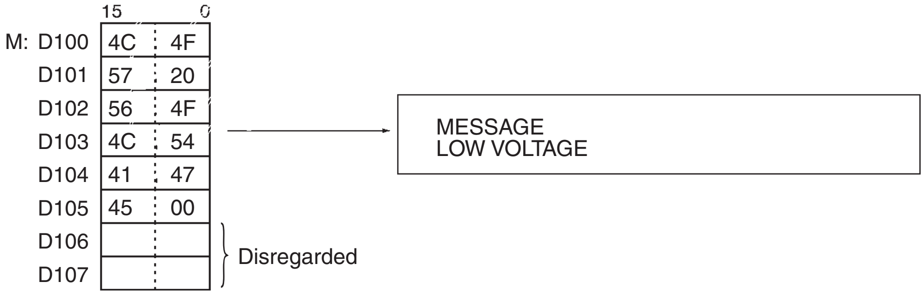
Generating a User-defined Error

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. The FALS Error Flag (A401.06) will be turned ON.
2. The corresponding error code (C11F) will be written to A400.
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 Peripheral Device.

Note If a message is not required, specify a constant for S.



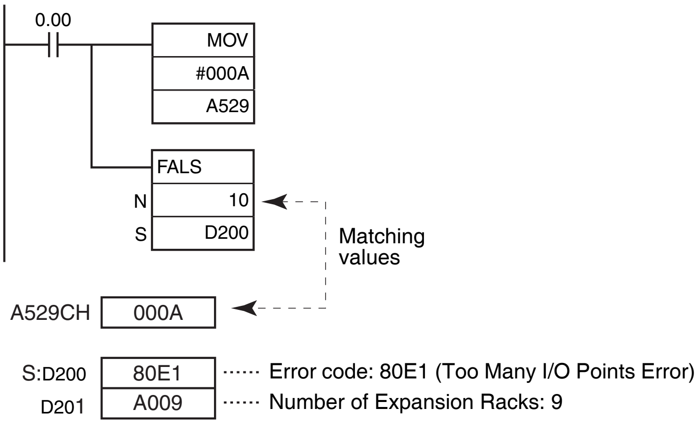


Note A400 will contain the error code of the most serious of all of the errors that have occurred, including non-fatal and fatal system errors, as well as errors generated by FAL(006) and FAL(007).

Generating a Non-fatal System Error

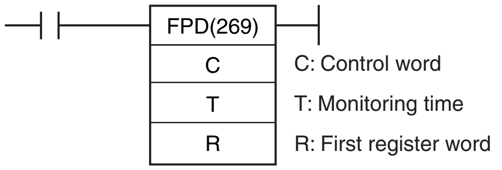
When CIO 0.00 is ON in the following example, FALS(007) will generate a Too Many I/O Points Error (too many Expansion Racks connected, 9 Racks in this case). In this case, dummy FAL number 10 is used and the corresponding value (000A hex) is stored in A529.

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 Programming Console indicating that a Too Many I/O Points Error has occurred.



FPD

Instruction	Mnemonic	Variations	Function code	Function
FAILURE POINT DETECTION	FPD	---	269	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.

	FPD
Symbol	

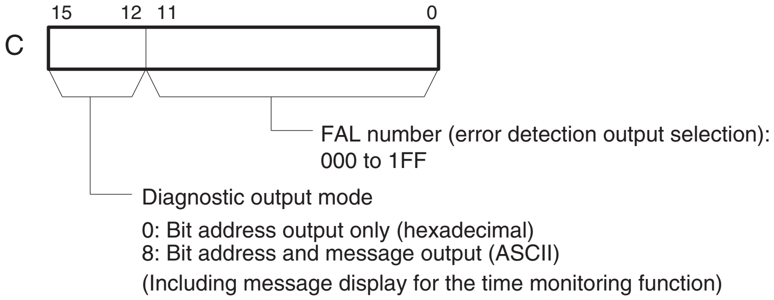
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	OK	OK	Not allowed	Not allowed

Operands

Operand	Description	Data type	Size
C	Control word	---	1
T	Monitoring time	---	1
R	First register word	---	Variable

C: 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 958.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
C	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---
T	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK				OK				
R											---							

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.

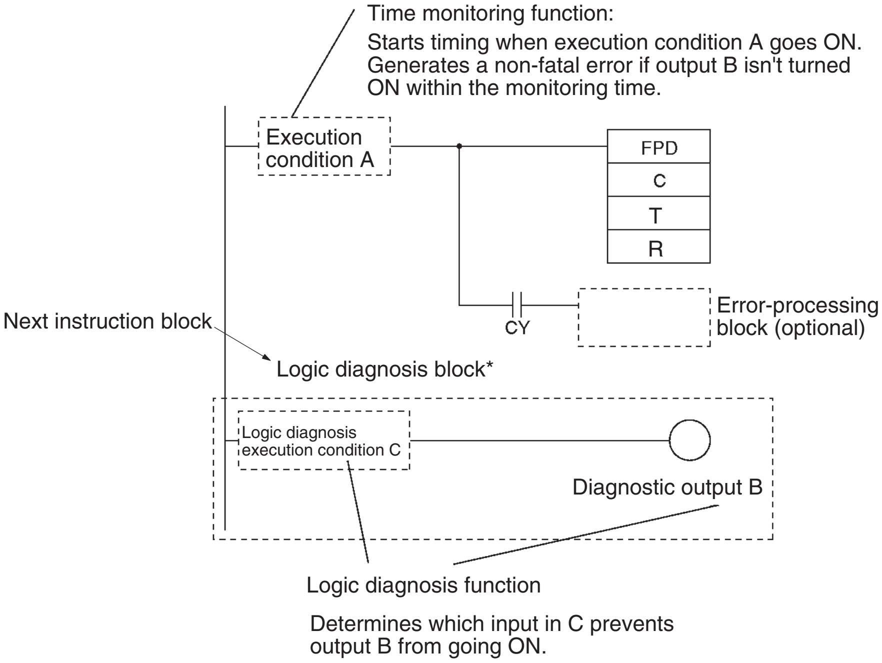
Related Auxiliary Area Words and Bits

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 A36001 to A39115 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).
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.

Function

FPD(269) performs time monitoring and logic diagnosis.

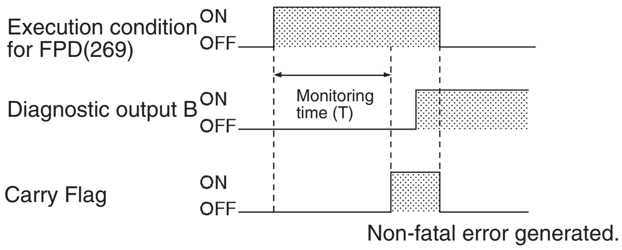
The next block is diagnosed at the following point.



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 1 The diagnostic output must go ON within the monitoring time.

2 The teaching function can be used set the monitoring time automatically.

Note 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.)

3 The FAL Error Flag (A402.15) will be turned ON. (PLC operation continues.)

4 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.)

5 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.)

6 The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).

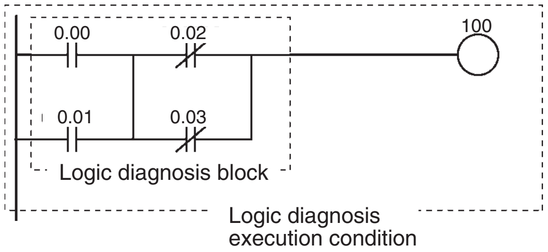
7 The ERR Indicator on the CPU Unit will flash.

8 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.

- 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.

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 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.

Note

- 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.
- The input circuit to be logically diagnosed should consist of multiple N.C. or N.O. conditions.
- 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.

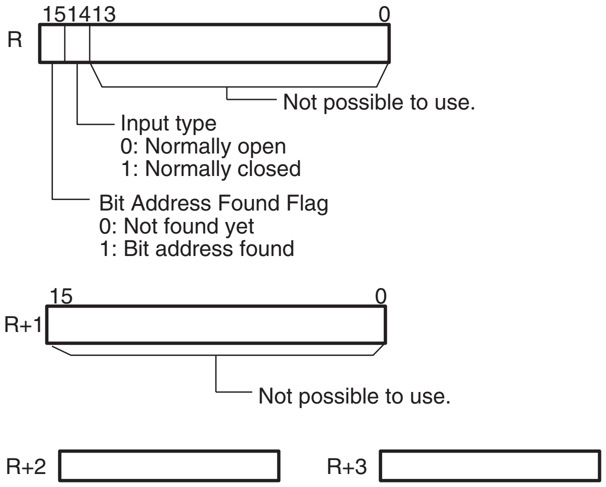
1. Bit address output mode (Leftmost digit of C = 0)
   Bit 15 of R (the Bit Address Found Flag) is turned ON when an input bit address has been found and bit 14 of R indicates whether the input is normally ON or normally OFF.
   The 8-digit hexadecimal PLC memory address of the input bit is output to 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.

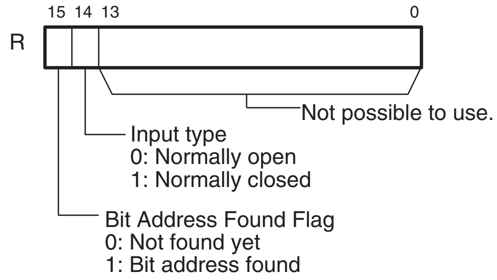
Bit Address Output (C=0□□□)

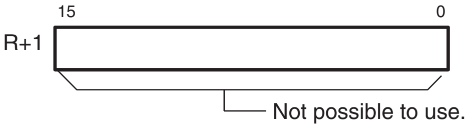
When the leftmost digit of C is set to 0, the 8-digit hexadecimal PLC memory address of the input bit is output to R+2 and R+3. R contains two flags which indicate whether an input bit has been found and whether it is used in a normally open or normally closed input condition.

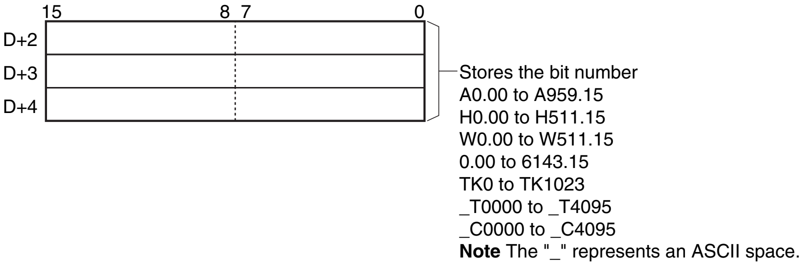


Bit Address and Message Output (C=8□□□)

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.

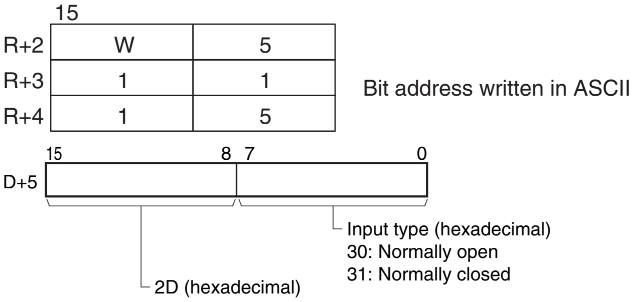






Stores the bit number
A0.00 to A959.15
H0.00 to H511.15
W0.00 to W511.15
0.00 to 6143.15
TK0 to TK1023
_T0000 to _T4095
_C0000 to _C4095
Note The "_" represents an ASCII space.

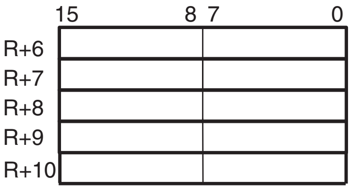
Register words R+2 through R+5 would have the following values for W51115.



	15	
R+2	W	5
R+3	1	1
R+4	1	5

D+6 to D+9

The user can store an ASCII message in register words R+6 to R+10. This message will be displayed on the Programming Device if a non-fatal error is generated by the time monitoring function. Mark the end of the message with the null character (00 hexadecimal).



Hint

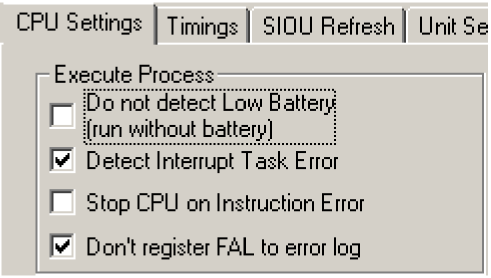
Disabling Error Log Entries of Non-fatal FPD(269) Errors

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 FAL(006) are not recorded in the Error Log.

Note Even though the error will not be recorded in the Error Log, the FAL Error Flag (402.15) 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 screen capture shows the PLC Setup setting from the CX-Programmer.



Execute Process

☐ Do not detect Low Battery (run without battery)

☑ Detect Interrupt Task Error

☐ Stop CPU on Instruction Error

☑ Don't register FAL to error log

- The following table shows the PLC Setup setting from the Programming Console.

Programming Console setting address	Word	129
	Bit	15
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)	

Note 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)

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.

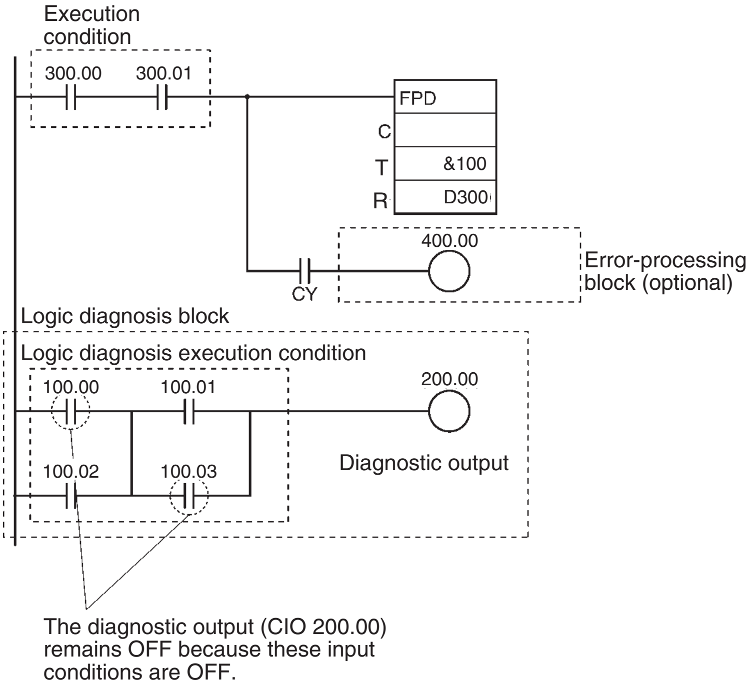
1. Turn ON the FPD Teaching Bit (A598.00).
2. FPD(269) will measure the time from the point when the execution condition for FPD(269) goes ON until the diagnostic output is turned ON.
3. If the measured time exceeds the monitoring time setting, a setting 1.5 times the measured time will be stored in T.

Note

- 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.

Example Programming

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 200.00) does not go ON because CIO 100.00 and CIO 100.03 remain OFF in the logic diagnosis execution condition.



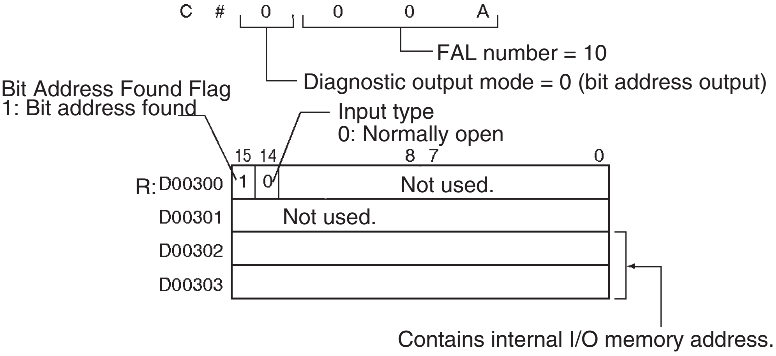
Time Monitoring Function

If the diagnostic output (CIO 200.00) 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.

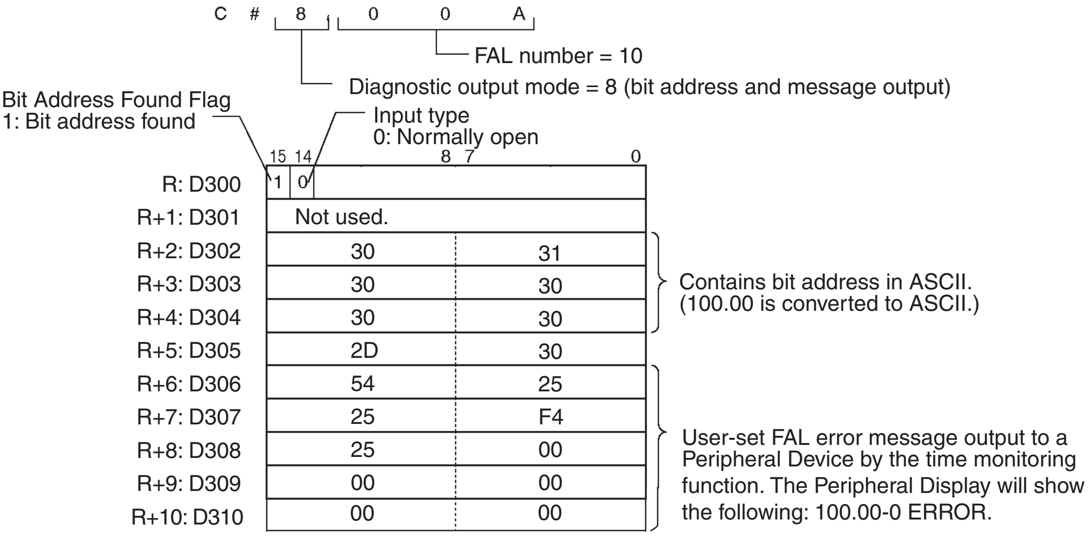
1. The Carry Flag is turned ON.
2. When the rightmost 3 digits of C specify an FAL number of 00A hex (10), the corresponding Executed FAL Number Flag (A360.10) will be turned ON, the corresponding error code (410A) is written in A400, and the FAL Error Flag (A402.15) is turned ON.

Logic Diagnosis Function (C=000A)

Since the leftmost digit of C is 0 (bit address output mode) the PLC memory address of CIO 100.00 is output to D303 and D302. (CIO 100.00 is on a higher instruction line than CIO 100.03.)

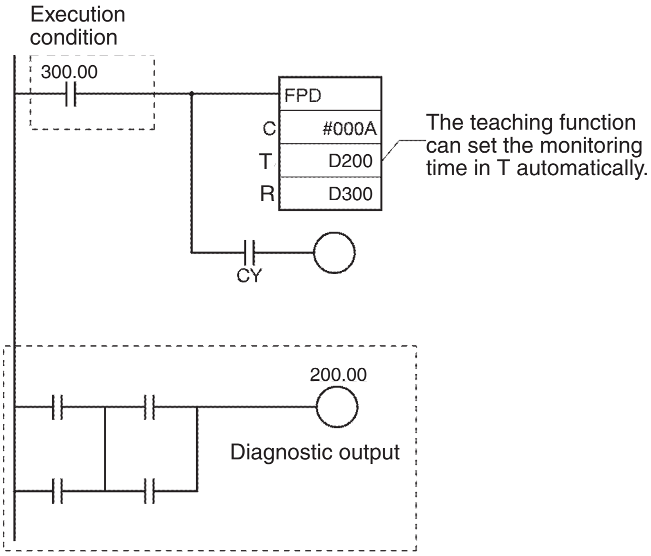


Since the leftmost digit of C is 8 (bit address and message output mode) the address of CIO 100.00 (100.00) is output to D302 through D304 in ASCII.

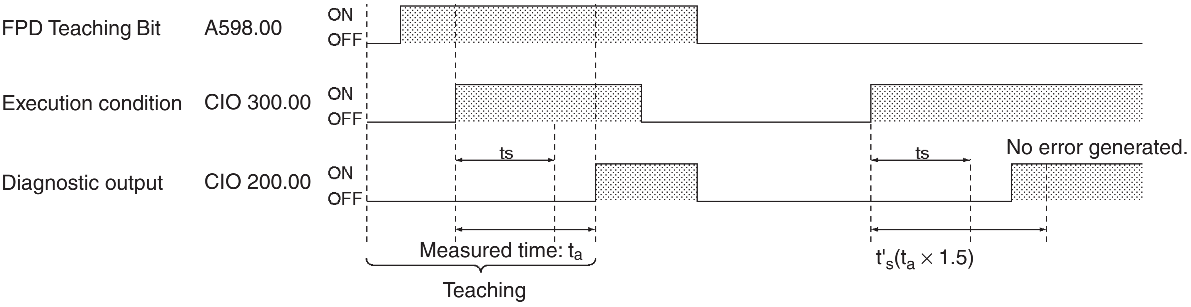


Setting the Monitoring Time with the Teaching Function

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 300.00) 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

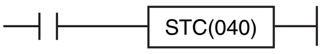
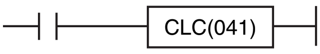
t'_s: New setting in T after teaching

(When $t_a > t_s$, $t'_s = t_a \times 1.5$)

Other Instructions

STC/CLC

Instruction	Mnemonic	Variations	Function code	Function
SET CARRY	STC	@STC	040	Sets the Carry Flag (CY).
CLEAR CARRY	CLC	@CLC	041	Turns OFF the Carry Flag (CY).

	STC	CLC
Symbol	 STC(040)	 CLC(041)

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

Operand	Description	Data type	
		STC	CLC
Error Flag	ER	Unchanged*1	Unchanged*1
Equals Flag	=	Unchanged*1	Unchanged*1
Carry Flag	CY	ON	ON
Negative Flag	N	Unchanged*1	Unchanged*1

*1 In CJ2, CS1-H, CJ1-H, CJ1M, and CS1D (for Single-CPU System) CPU Units, these Flags are left unchanged.
In CS1 and CJ1 CPU Units, these Flags are turned OFF.

Function

● STC

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.

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.

● CLC

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.

+C(402), +CL(403), +BC(406), +BCL(407), -C(412), -CL(413), -BC(416), and -BCL(417) 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.

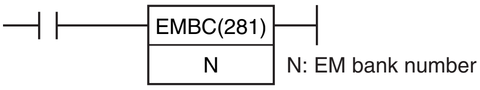
ROL(027), ROLL(572), ROR(028), and RORL(573) make use of the Carry Flag in their rotation shift operations. When using any of these instructions, use STC(040) and CLC(041) to set and clear the Carry Flag.

Hint

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.

EMBC

Instruction	Mnemonic	Variations	Function code	Function
SELECT EM BANK	EMBC	@EMBC	281	Changes the current EM bank.

	EMBC
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Ok	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	EM bank number	UINT	1

N: EM Bank Number

Specifies the new EM bank number in decimal (0 to 24) or hexadecimal (0000 to 0018).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---

Flags

Operand	Description	Data type
Error Flag	ER	• ON if N specifies a non-existent EM bank number. (This error will occur if the specified EM bank has been registered as file memory in the PLC Setup.) • ON if automatic address allocation has been set for the specified EM bank (CJ2 CPU Units only). • ON if the EM Area force-set/reset function is enabled (CJ2 CPU Units only). • ON if EMBC(281) is executed in an interrupt task for a CJ2 CPU Unit with high-speed interrupt function enabled. • OFF in all other cases.

Related Auxiliary Area Words and Bits

Name	Address	Operation
Current EM Bank	A301	Contains the current EM bank number in hexadecimal

Function

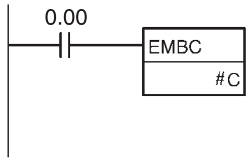
EMBC(281) changes the current EM (Extended Data Memory) bank to the one indicated by the EM bank number (N). At the same time, the new EM bank number is output to A301.

Note
- The current EM bank number changed in a cyclic task is retained when operation is switched between tasks. For example, if EMBC(281) is used in task 1 to change the current EM bank from bank B to bank C, bank C will remain the current EM bank for all cyclic tasks even when operation is switched to task 2.
- The current EM bank number changed in an interrupt task is valid only during execution of the interrupt in which it was changed. The previous EM bank number will be returned to once execution of the interrupt task has been completed.

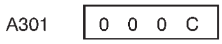
Hint

- There are banks available in the EM Area and there are 32,768 words (E00000 to E32767) in each bank. EM addresses can be identified in the two following ways. EMBC(281) must be used to change the current EM bank if the first method is used.
  1) EM addresses can be specified without the bank number, i.e. E00000 to E32767, to indicate addresses in the current EM bank.
  2) EM addresses can be specified with the bank number, i.e. En_00000 to En_32767 (n = bank number), to indicate addresses in a particular EM bank.

Example Programming

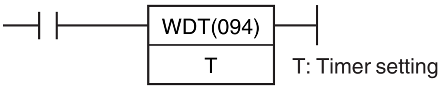


When CIO 0.00 turns ON in the following example, the current EM bank number is changed to bank C and the new bank number (000C hex) is output to A301.



WDT

Instruction	Mnemonic	Variations	Function code	Function
EXTEND MAXIMUM CYCLE TIME	WDT	@WDT	094	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.

	WDT
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Ok	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
T	Timer setting	Constants only	1

T: Timer setting

Specifies the watchdog timer setting between 0000 and 0F9F hexadecimal or between &0000 and &3999 decimal.

● Operand Specifications

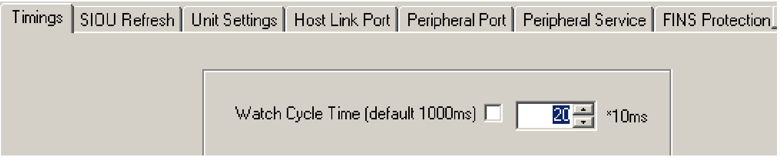
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
T	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Flags

Operand	Description	Data type
Error Flag	ER	• ON if the watchdog timer setting exceeds 40 seconds. • OFF in all other cases.

Related PLC Setup Settings

● CX-Programmer settings



Watch Cycle Time (default 1000ms) ☐ 20 *10ms

● Programming console settings

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)

Note
- 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.
- 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.

Related Auxiliary Area Words and Bits

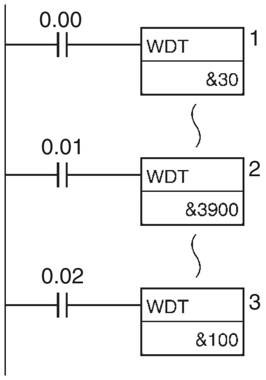
Name	Address	Operation
Cycle Time Too Long Flag	A401.08	ON when the present cycle time exceeds the maximum cycle time (watch cycle time) set in the PLC Setup. This is a fatal error which causes program execution to stop.
Maximum Cycle Time	A262 and A263	These words contain the maximum cycle time in 32-bit 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.

Function

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).

When it is likely that the cycle time will increase due to a temporary increase in processing data, this instruction can be used to prevent a cycle time error.

Example Programming



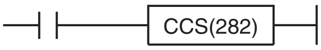
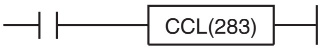
Operation of WDT(094)

The default maximum cycle time (1,000 ms) is used in this example.

- 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.
- 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.
- 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.

CCS/CCL

Instruction	Mnemonic	Variations	Function code	Function
SAVE CONDITION FLAGS	CCS	@CCS	282	Saves the current status of the Condition Flags in a separate area within the CPU Unit.
LOAD CONDITION FLAGS	CCL	@CCL	283	Restores the status of the Condition Flags that were saved in a separate area within the CPU Unit by CCS(282).

Symbol	CCS	CCL
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

● CCS

There are no flags affected.

● CCL

Changes to the value that was read.

Function

● CCS

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 <=.

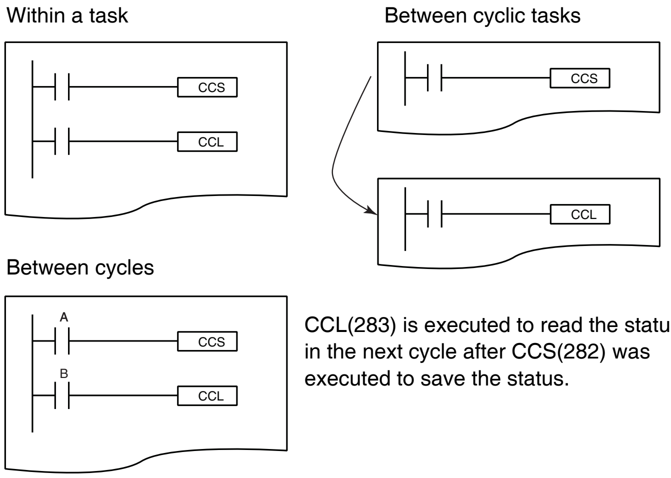
Note When CCS(282) is executed, it overwrites the previous Condition Flag information that was saved.

● CCL

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 <=.

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



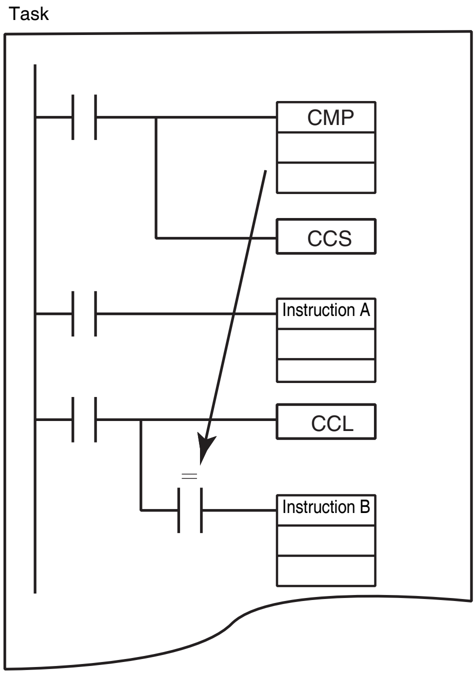
CCL(283) is executed to read the status in the next cycle after CCS(282) was executed to save the status.

The status of the Condition Flags cannot be saved/loaded between a cyclic task and interrupt task.

Hint

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.



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.)

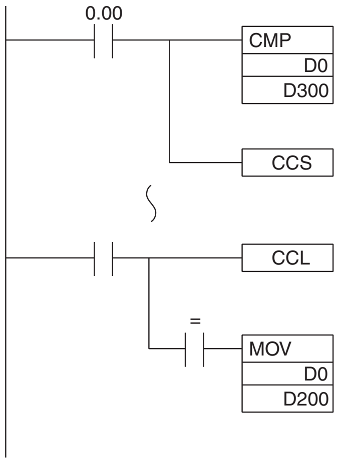
Preserves the status of the Condition Flags in a separate location in the CPU Unit.

Restores the status of the Condition Flags.

The Equals Flag will reflect the result of the COMPARE instruction, not the result of instruction A.

- 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.

Example Programming



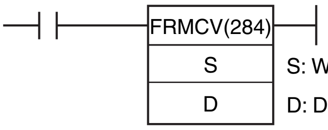
When CIO 0.00 is ON in the following example, CMP(020) will compare the contents of D0 and D300 and those results will be preserved by CCS(282).

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.

FRMCV

Instruction	Mnemonic	Variations	Function code	Function
CONVERT ADDRESS FROM CV	FRMCV	@FRMCV	284	Converts a CV-series PLC memory address to its corresponding CS/CJ-series PLC memory address.

Symbol	FRMCV
	 S: Word containing the CV-series PLC memory address D: Destination Index Register

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Ok	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Word containing the CV-series PLC memory address	UINT	1
D	Destination Index Register	IR*1 only	2

*1 Only an index register (IR0 to 15) can be specified in D.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	Ok	OK	OK	OK	OK	---	OK	---	---	---	---
D	---	---	---	---	---	---	---	---	---	---	---	---	OK	---				

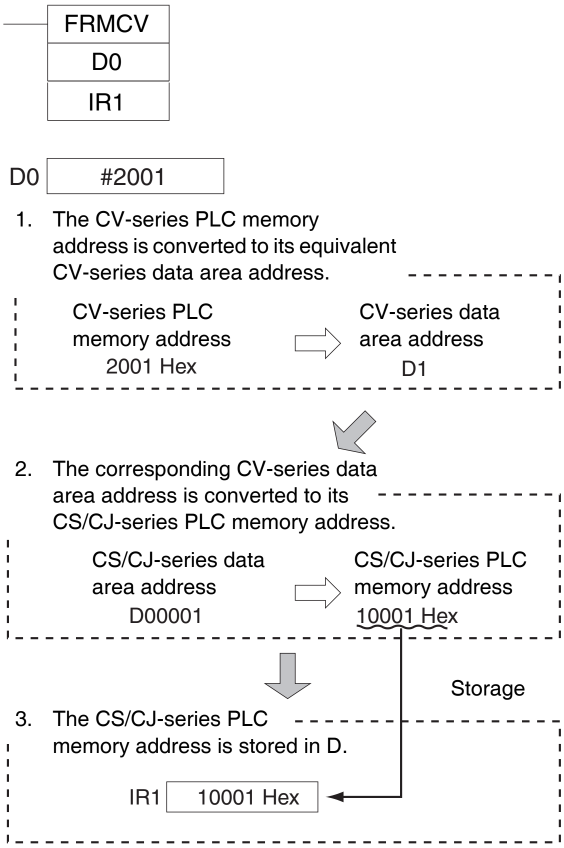
Flags

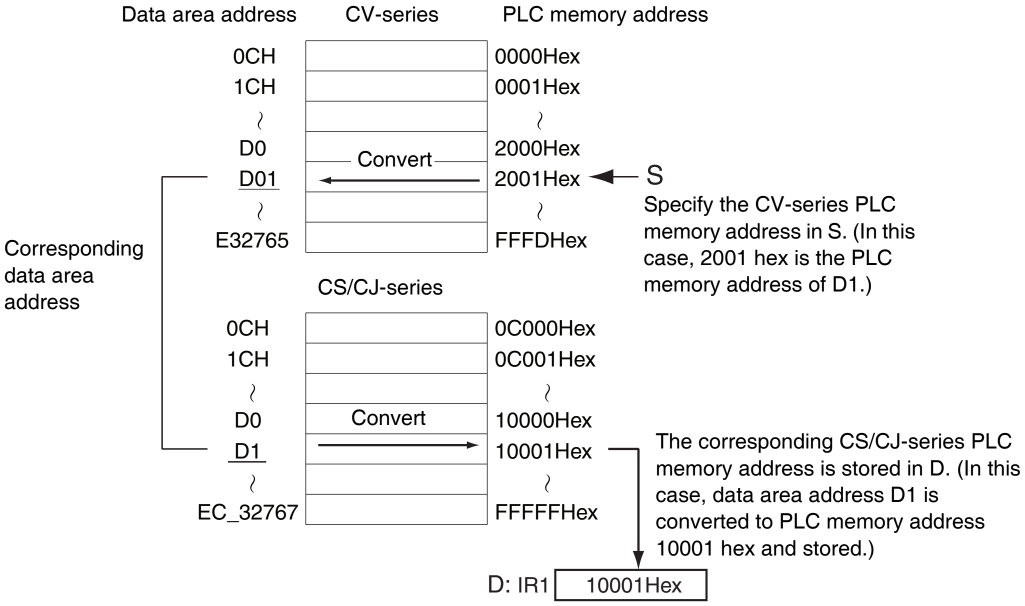
Operand	Description	Data type
Error Flag	P_ER	• ON if S specifies one of the following PLC memory addresses that do not exist in the CS/CJ-series: • TR Area (09FF hex) • CPU Bus Link (G) Area (0A00 to 0AFF hex) • SFC Areas (0D00 to 0E3F hex) • OFF in all other cases.

Function

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 CS/CJ-series PLC memory address that corresponds to the same CV-series data area address.
3. The CS/CJ-series PLC memory address is output to D. (An index register (IR0 to IR15) must be specified for D.)





Note If there is no CS/CJ-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.

Hint

- 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.

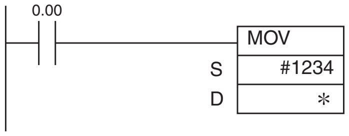
- The FRMCV(284) instruction can be used to convert a CV-series program with the following two kinds of programming for use in a CS/CJ-series PLC. See the Examples later in this section for an example.
  1. When using indirect binary mode DM addressing (*DM)
     (when indirectly specifying a data area 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))

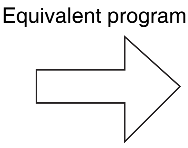
Example Programming

● Example 1: Converting a CV-series Program with *DM Indirect Binary Mode DM Addressing

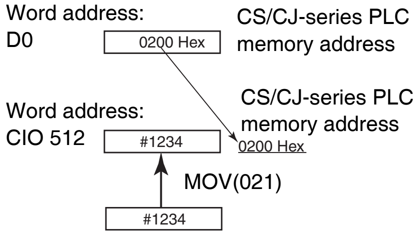
In this FRMCV(284) example, a DM word is specified in S, the PLC memory address there is stored in an Index Register, and the Index Register is used for indirectly addressed.

- CV-series program
  (Program using indirect DM binary mode addressing)

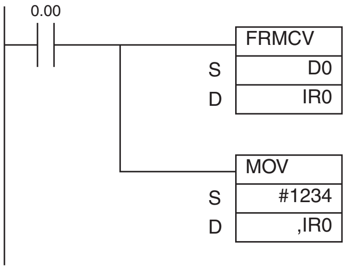




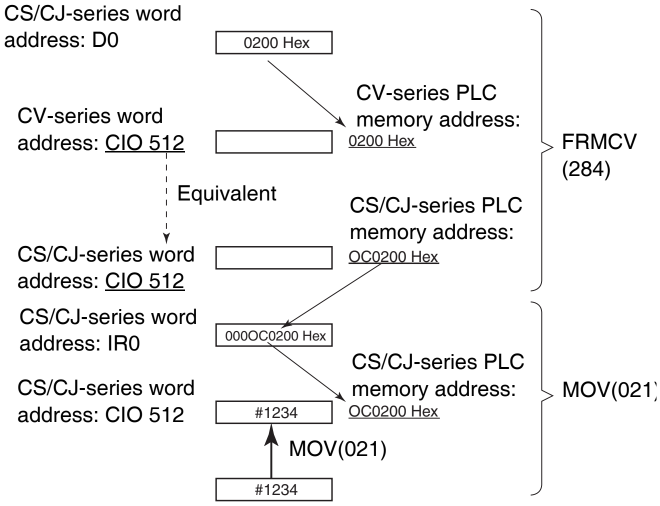
PLC Setup
Indirect DM data:
When indirect DM addresses are in binary, the content of the DM word is treated as a PLC memory address and specifies the corresponding address in I/O memory.
In this case, the value in D0 is 0200 hex. The corresponding data area address is CIO 512, so #1234 is transferred to CIO 512.



- CS/CJ-series program



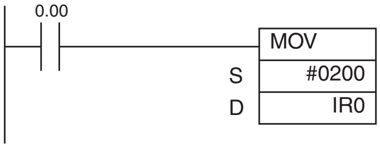
In this case, the value in D0 is 0200 hex. The corresponding CV-series data area address is CIO 512. The CS/CJ-series PLC memory address for CIO 512 is 0C200 hex, so this value is stored in IR0. The destination operand in MOV(021) indirectly addresses the content of IR0, so #1234 is transferred to CIO 512.



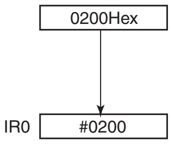
Example 2: Converting a CV-series Program with PLC Memory Addresses Stored directly in Index Registers

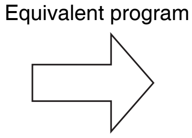
In this FRMCV(284) example, the CV-series PLC memory address is specified directly in S.

- CV-series program
  (Program using PLC memory
  addresses stored directly in IR)

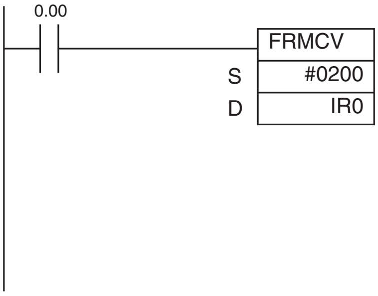


In this case, the PLC memory address 0200 hex is stored in Index Register IR0.

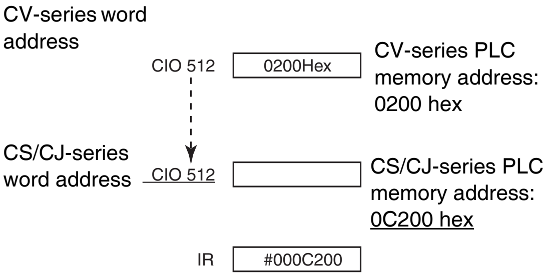




- CS/CJ-series program

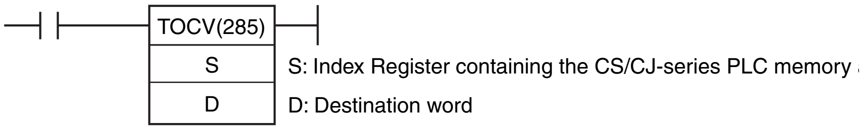


In this case, the CV-series PLC memory address 0200 hex corresponds to CIO 512. The CS/CJ-series PLC memory address for CIO 512 is 0C200 hex, so this value is stored in IR0.



TOCV

Instruction	Mnemonic	Variations	Function code	Function
CONVERT ADDRESS TO CV	TOCV	@TOCV	285	Converts a CS/CJ-series PLC memory address to its corresponding CV-series PLC memory address.

Symbol	TOCV
	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	Ok	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Index Register containing the CS/CJ-series PLC memory address	IR*1 only	2
D	Destination word	UINT	1

*1 Only an index register (IR0 to 15) can be specified in D.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	---	---	---	---	---	---	---	---	---	---	OK	---	OK	---	---	---	---	---
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK				

Flags

Operand	Description	Data type
Error Flag	P_ER	• ON if S specifies a PLC memory address that does not exist in the CV-series PLCs. • OFF in all other cases.

Note

- 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

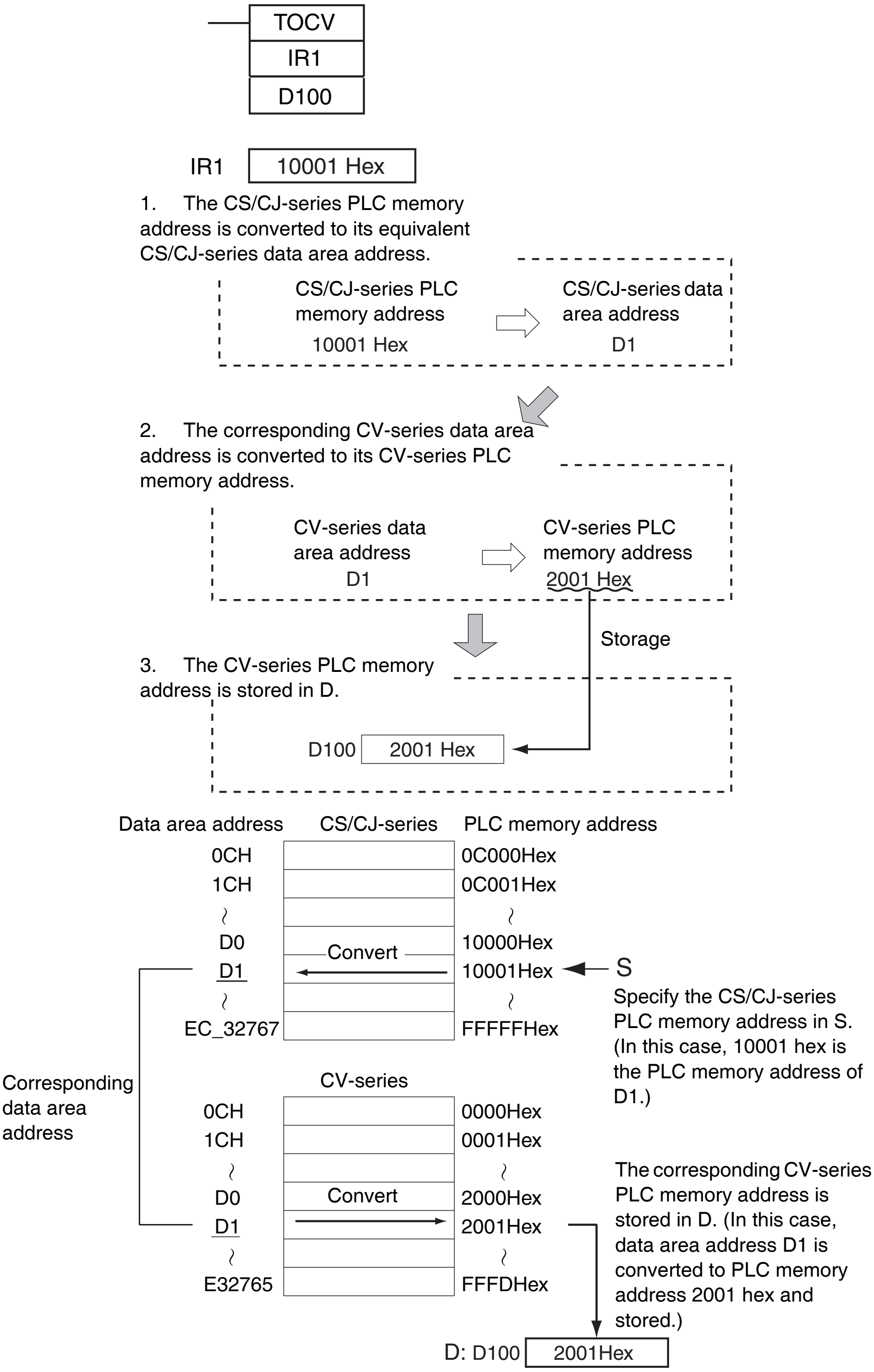
Area or addresses	PLC memory addresses
Task Flag Area	0000 B800 to 0000 B801 hex
A10000 to A11535	0000 B200 to 0000 B7FF hex
CIO 2556 to CIO 6143	0000 C9FC to 0000 D7FF hex
T1024 to T4095	0000 BE40 to 0000 BEFF hex and 0000 E400 to 0000 EFFF hex
C1024 to C4095	0000 BF40 to 0000 BFFF hex and 0000 F400 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
EM bank specification	0001 8000 to 000F 7FFF hex
E32766 to D32767	000F FFFE to 000F FFFF hex

- 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.

Function

When the execution condition is ON, TOCV(285) executes the following operations.

1. The CS/CJ-series PLC memory address specified in S is converted to its equivalent CS/CJ-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 CS/CJ-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 CS/CJ-series PLC memory address for D1.



Note If there is no CV-series equivalent to the specified CS/CJ-series PLC memory address, an error will occur, the Error Flag will be turned ON, and the address will not be converted.

Hint

After this instruction is executed, the basic methods of use are as follows.

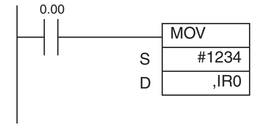
1. The CV-series PLC memory address data stored by TOCV(285) can be transferred to a CV-series PLC using CX-Programmer.
2. The same data area address that was used in the CS/CJ-series program can be specified in the CV-series program by using indirect Index Register addressing (",IR" prefix) or indirect binary mode DM addressing (*DM).

Example Programming

Converting a CS/CJ-series Program with Indirect Index Register Addressing

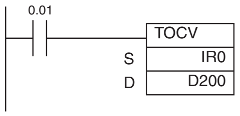
1. In this TOCV(285) example, an Index Register is specified in S. The CS/CJ-series PLC memory address in that Index Register is converted to its CV-series equivalent.
2. The CV-series PLC memory address is transferred to the specified data area address.
3. Use the CV-series PLC memory address in the CV-series program.

- CS/CJ-series program (Program using indirect Index Register addressing)



In this case, IR0 contains 10001 hex. The data area address corresponding to PLC memory address 10001 hex is D1, so #1234 is transferred to D1.

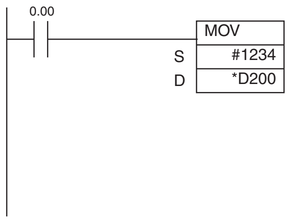
- CS/CJ-series program



In this case, IR0 contains 10001 hex. Since the data area address corresponding to CS/CJ-series PLC memory address 10001 hex is D1, TOCV(285) stores the CV-series PLC memory address for D1 (2001 hex) in destination word D200.

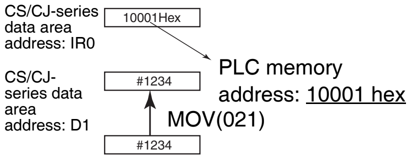
Transfer contents of D200 to CV-series.

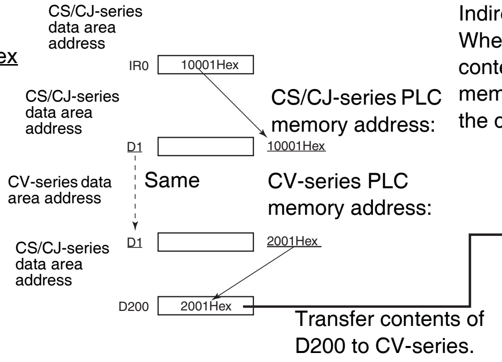
- CV-series program

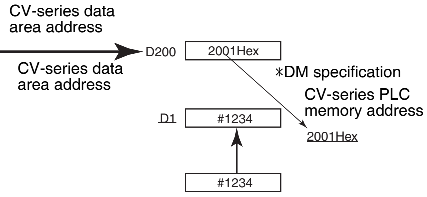


In the CV-series PLC, the destination of the MOV(021) instruction is indirectly addressed (in binary mode) through D200, so #1234 is transferred to D1.

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.

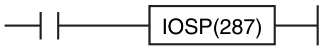






IOSP/IORS

Instruction	Mnemonic	Variations	Function code	Function
DISABLE PERIPHERAL SERVICING	IOSP	@IOSP	287	Disables peripheral servicing during program execution in Parallel Processing Mode or Peripheral Servicing Priority Mode.
ENABLE PERIPHERAL SERVICING	IORS	---	288	Enables the peripheral servicing during program execution in Parallel Processing Mode that was disabled by IOSP(287), the DISABLE PERIPHERAL SERVICING instruction.

	IOSP	IORS
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Flags

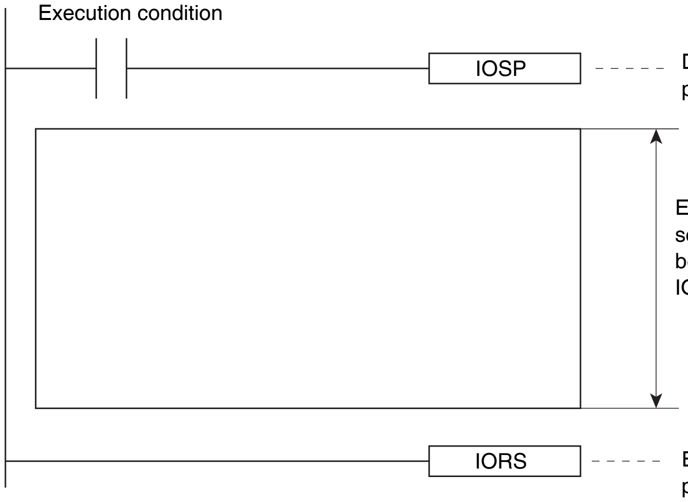
Operand	Description	Data type
Error Flag	P_ER	• ON if IOSP(287) is executed in an interrupt task. • OFF in all other cases.

Function

● IOSP

Use IOSP(287) in a cyclic task in Parallel Processing Mode (with Synchronous or Asynchronous Memory Access) to disable the following kinds of peripheral servicing. Peripheral servicing will be enabled again when IORS(288), the ENABLE PERIPHERAL SERVICING instruction, is executed.

- Event servicing with Special I/O Units
- Event servicing with CPU Bus Units
- Peripheral Port servicing
- RS-232C Port servicing
- Event servicing with Inner Boards (CS-series only)
- Event servicing (including background instruction processing) that uses a communications port number, i.e., an internal logical port.



Disables execution of peripheral servicing.

Execution of peripheral servicing is disabled between IOSP(287) and IORS(288).

Enables execution of peripheral servicing.

When peripheral servicing has been disabled with IOSP(287), it will remain disabled until IORS(288) is executed, END(001) is executed, or PLC operation is stopped.

Note
- IOSP(287) cannot be executed in an interrupt task. An error will occur and the Error Flag will be turned ON if IOSP(287) is executed in an interrupt task.
- IOSP(287) cannot disable peripheral servicing in more than one task. If it is necessary to disable peripheral servicing in more than one task, program IOSP(287) separately in each task.

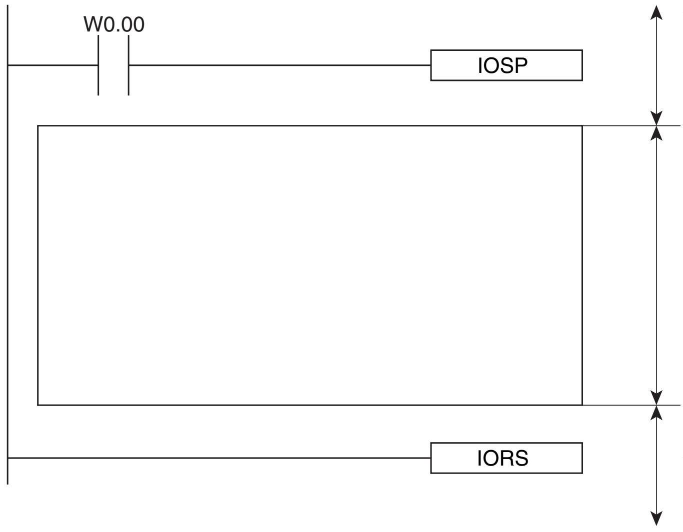
● IORS

Use IORS(288) in a cyclic task to release the prohibition on peripheral servicing by IOSP(287), the DISABLE PERIPHERAL SERVICING instruction.

Note 1 It is not necessary to program IORS(288) with an execution condition.

2 IORS(288) cannot be executed in an interrupt task. An error will occur and the Error Flag will be turned ON if IORS(288) is executed in an interrupt task.

Example Programming



When the PLC is in Parallel Processing Mode, peripheral servicing is executed in parallel.

When W0.00 is ON, execution of peripheral servicing is disabled between IOSP(287) and IORS(288).

Enables execution of peripheral servicing. When the PLC is in Parallel Processing Mode, peripheral servicing is executed in parallel.

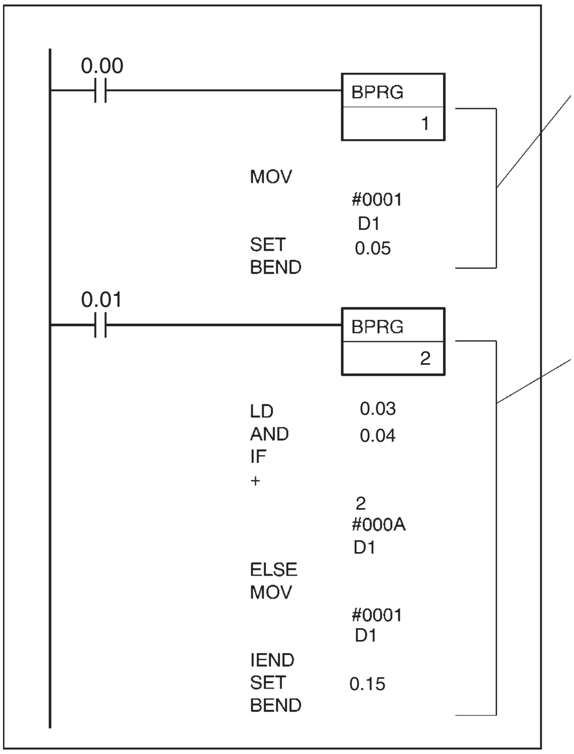
Block Programming Instructions

Block Programming Instructions

Block Programs

- Up to 128 block programs within the overall user program (all tasks) with the CS/CJ-series. 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.

Example: The following example shows two block programs.



Block program area No. 1

When CIO 0.00 is ON, the contents of block program 1 will be executed. The MOV(021) and SET instructions will be executed unconditionally and the block program will end.

Block program area No. 2

When CIO 0.01 is ON, the contents of block program 2 will be executed. If CIO 0.03 and CIO 0.04 are both ON, the binary addition will be performed (CIO 0.02 + #000A→D0.01).
If one or both of these bits is OFF, #0001 will be moved to D1. CIO 0.15 will then be set unconditionally and the block program will end.

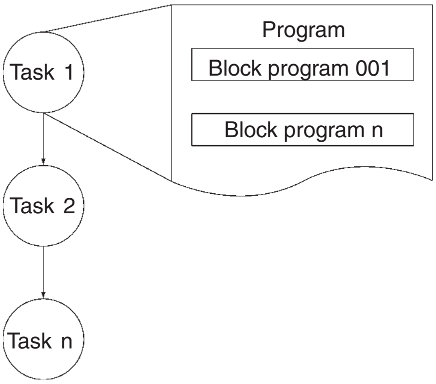
Tasks and Block Programs

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.

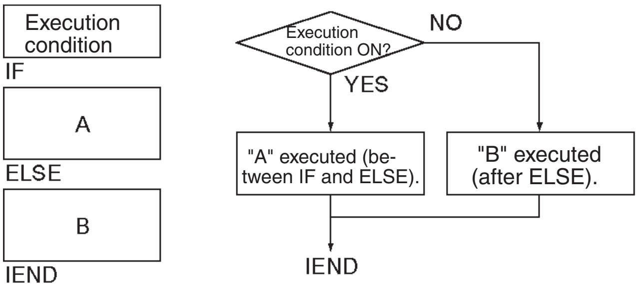
Block programs can be used either within cyclic tasks or interrupt tasks. Each block program number from 0 to 127 can be used only once and cannot be use again, even in a different task.



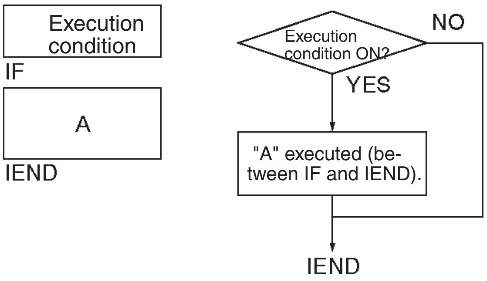
Using Block Programming Instructions

- 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.

● Instructions with usage restrictions

(Instructions used only as execution conditions IF, WAIT, EXIT, and LEND)

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   1.00
TST   D1        #0010
IF
```

Used as execution condition for IF.

× **Bad Example**

```
LD    0.00
AND   1.00
TST   D1        #0010
MOV   #0000     10
```

Cannot be used as execution condition for MOV(021).

Instructions Not Applicable in Block Programs

The instructions listed in the following table cannot be used within block programs.

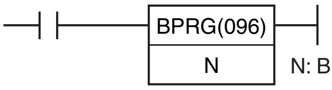
● Instructions Not Applicable in Block Programs

Instruction group	Mnemonic	Name	Alternative
Sequence Output Instructions	OUT	OUTPUT	Use SET and RSET.
	OUT NOT	OUTPUT NOT	
	DIFU(013)	DIFFERENTIATE UP	None
	DIFD(014)	DIFFERENTIATE DOWN	None
	KEEP(011)	KEEP	None
Sequence Control Instructions	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 and TIMX(550)	HUNDRED-MS 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) and TIMHX(551)	TEN-MS TIMER	
	TMHH(540) and TMHHX(552)	ONE-MS TIMER	
	TIMU(541) and TIMUX(556)	TENTH-MS TIMER	
	TIMUH(544) and TIMUHX (557)	HUNDREDTH-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	
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

Note JMP and JME instructions can be used. However, an execution condition is not used; the program jumps to JME unconditionally. CJP and JME instructions can also be used. These are entered after an execution condition and the program jumps to JME as usual based on the execution condition.

BPRG/BEND

Instruction	Mnemonic	Variations	Function code	Function
BLOCK PROGRAM BEGIN	BPRG	---	096	Indicates the beginning of the block program area specified by number.
BLOCK PROGRAM END	BEND	---	801	Indicates the end of the block program area.

	BPRG	BEND
Symbol	 BPRG(096) N N: Block program number	BEND(801)

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	Not allowed	OK	OK	OK	Not allowed

Operands

Operand	Description	Data type	Size
N	Block program number	Number	1

N: Block Program Number

The block program number must be between 0 and 127 decimal.

● Operand Specifications

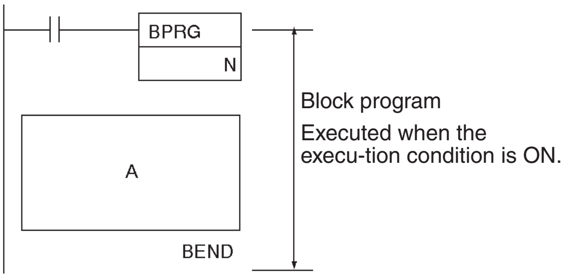
Area		Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
		CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
BPRG	N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Flags

Name	Label	Operation	
		BPRG	BEND
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.	• ON if a block program is not being executed. • OFF in all other cases.

Function

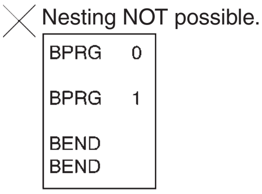
BPRG(096) executes the block program with the block number designated in N, i.e., the one immediately after it and ending with BEND(801). All instructions between BPRG(096) and BEND(801) are executed with ON execution conditions (i.e., unconditionally).



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.

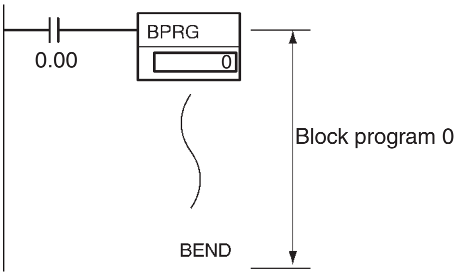
Precautions

- Each block program number can be used only once within the entire user program.
- Block programs cannot be nested.

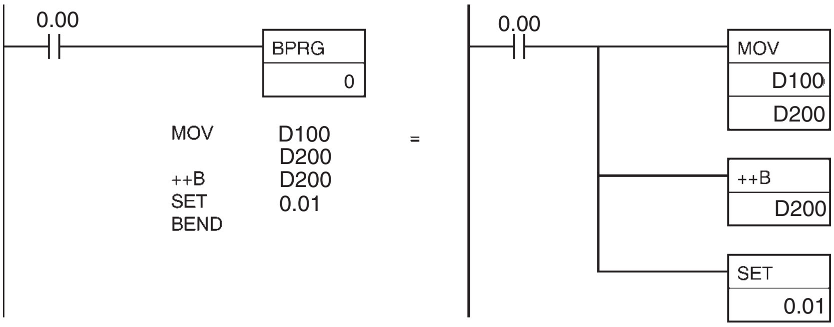


- If the block program is in an interlocked program section and the execution condition for IL(002) is OFF, the block program will not be executed.
- BPRG(096) and the corresponding BEND(801) must be in the same task.

Example Programming



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).

BPPS/BPRS

Instruction	Mnemonic	Variations	Function code	Function
BLOCK PROGRAM PAUSE	BPPS	---	811	Pause the specified block program from another block program.
BLOCK PROGRAM RESTART	BPRS	---	812	Restart the specified block program from another block program.

Symbol	BPPS	BPRS
	BPPS(811) N N: Block program number	BPRS(812) N N: Block program number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	OK*1	Not allowed

*1 BPRG(096) and BPRS(812) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

Operand	Description	Data type	Size
N	Block program number	Number	1

N: Block Program Number

The block program number must be between 0 and 127 decimal.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

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.

Note 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).

Function

● BPPS

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.

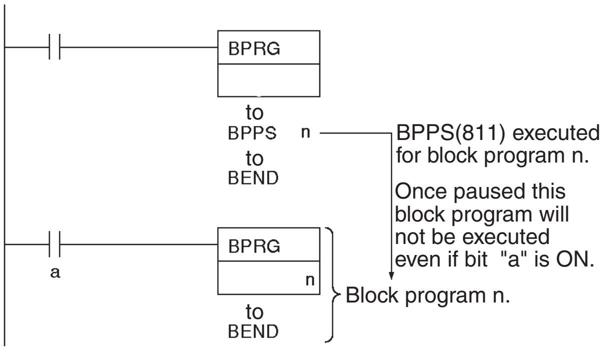
Note BPPS(811) can be used to pause the block program that contains it. When the block program is then restarted using BPRS(812) from another block program, the paused block program will restart from the next instruction after BPPS(811).

● BPRS

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.

Hint

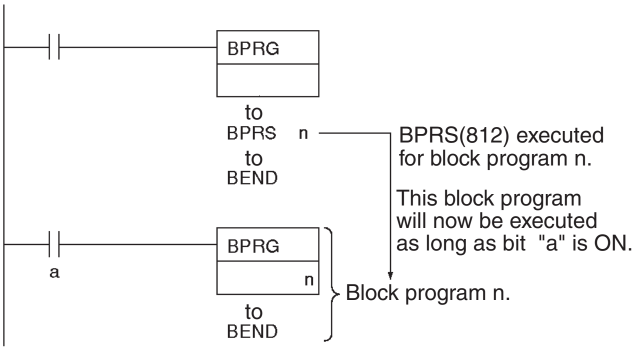
- If a paused block program contains TIMW(813), TIMWX(816), TMHW(815), or TMHWX(817), the PV of the timer will continue to elapse even while the block program is paused.



BPPS(811) executed for block program n.

Once paused this block program will not be executed even if bit "a" is ON.

Block program n.

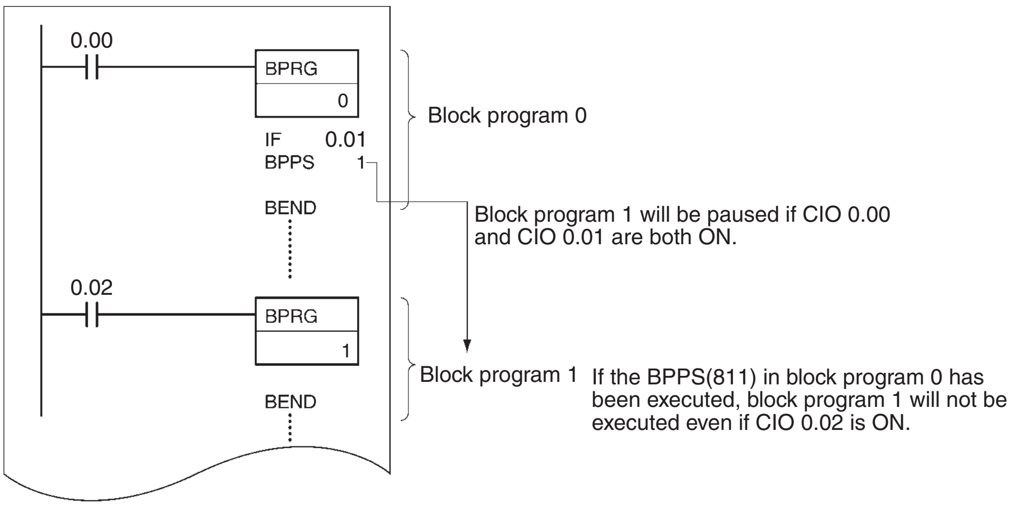


BPRS(812) executed for block program n.

This block program will now be executed as long as bit "a" is ON.

Block program n.

Example Programming



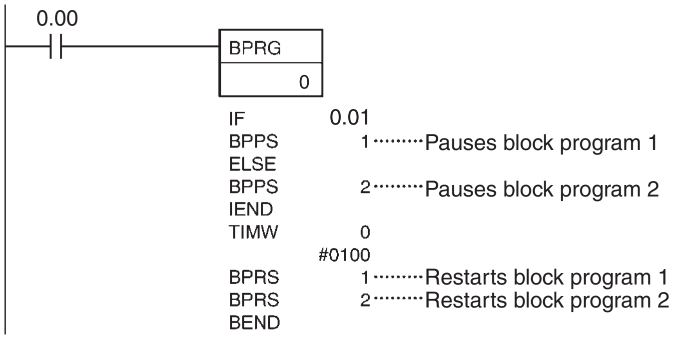
Block program 0

Block program 1 will be paused if CIO 0.00 and CIO 0.01 are both ON.

Block program 1

If the BPPS(811) in block program 0 has been executed, block program 1 will not be executed even if CIO 0.02 is ON.

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.



Pauses block program 1

Pauses block program 2

Restarts block program 1

Restarts block program 2

Address	Instruction	Operands
000200	LD	0.00
000201	BPRG(096)	0
000202	IF(802)	0.01
000203	BPPS(811)	1
000204	ELSE(803)	---
000205	BPPS(811)	2
000206	IEND(804)	---
000207	TIMW(803)	0000
		#0100
000208	BPRS(812)	1
000209	BPRS(812)	2
000210	BEND(801)	---

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.

EXIT/EXIT NOT

Instruction	Mnemonic	Variations	Function code	Function
CONDITIONAL BLOCK EXIT	EXIT	---	806	When the execution condition is ON (when an operand is not specified), or when the specified bit is ON (when an operand is specified), the instructions from the EXIT instruction to the BEND instruction are not executed and the block program is exited.
CONDITIONAL BLOCK EXIT NOT	EXIT NOT	---	806	When the specified bit is OFF, the instructions from the EXIT NOT instruction to the BEND instruction are not executed and the block program is exited.

Symbol	Operation without an operand (operation by execution condition)	Operation with an operand
	Execution condition EXIT(806)	EXIT(806) B B: Bit operand EXIT NOT(806) B B: Bit operand

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	OK*1	Not allowed

*1 EXIT(806) and EXIT NOT(806) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

Operand	Description	Data type	Size
B	Bit operand	BOOL	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	OK	OK	OK*1	OK*1	---	---	---	---	---	OK	OK	OK	OK	---

*1 CJ2 CPU Units only.

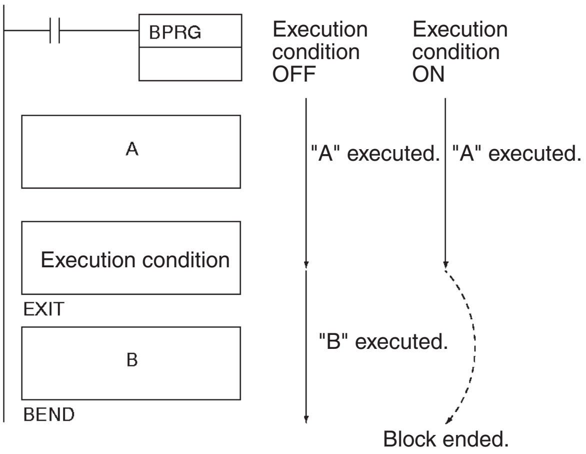
Flags

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.

Function

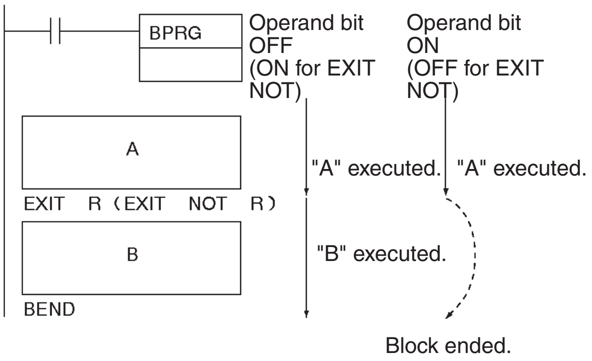
● Operation without an Operand

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

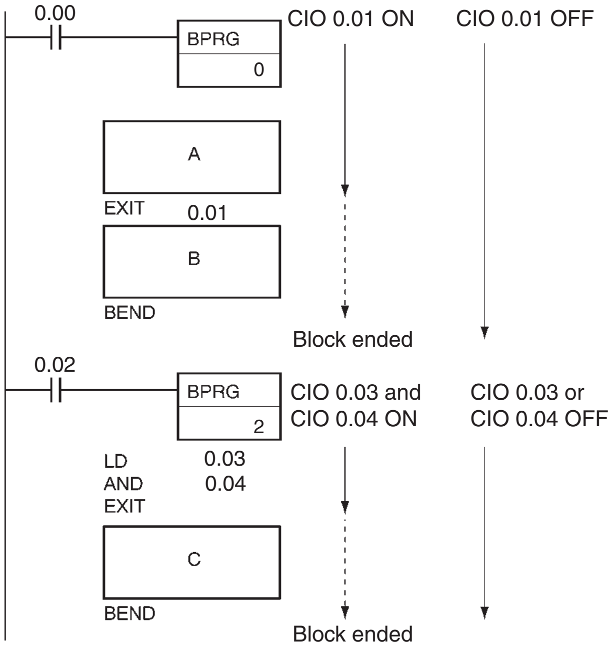
If the operand bit, B, is OFF for EXIT(806) the rest of the block program will be executed normally. If the operand bit is ON for EXIT(806), the rest of the instructions in the block program through BEND(801) will not be executed. For EXIT NOT(806), the rest of the block program will be executed for if the operand bit is ON and skipped if the operand bit is OFF.



Example Programming

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(096)	0
⋮	A	⋮
000210	EXIT(806)	0.01
⋮	B	⋮
000220	BEND(811)	---
000221	LD	0.02
000222	BPRG(096)	2
000223	LD	0.03
000224	AND	0.04
000225	EXIT(806)	---
⋮	C	⋮
000230	BEND(801)	---

IF/IF NOT/ELSE/IEND

Instruction	Mnemonic	Variations	Function code	Function
Branching	IF	---	802	When the execution condition (when an operand is not specified) or the specified bit (when an operand is specified) is OFF, the next and following instructions are executed. When ON, instructions up to ELSE are disregarded.
	IF NOT	---	802	When the specified bit is ON, the next and following instructions are executed. When OFF, instructions up to ELSE are disregarded.
	ELSE	---	803	Indicates the beginning of the block that is executed when IF is false.
	IEND	---	804	Indicates the end of the conditional branch block area.

Symbol	Operation without an IF instruction operand (operation by execution condition)	Operation with an IF (IF NOT) instruction operand
	Execution condition IF(802) : ELSE(803) : IEND(804)	IF(802) B B: Bit operand IF NOT(802) B B: Bit operand : ELSE(803) B B: Bit operand : IEND(804) B B: Bit operand

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	OK*1	Not allowed

*1 IF(802), ELSE(803), and IEND(804) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

Operand	Description	Data type	Size
B	Bit operand	BOOL	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	OK	OK	OK*1	OK*1	---	---	---	---	---	OK	OK	OK	OK	---

*1 CJ2 CPU Units only.

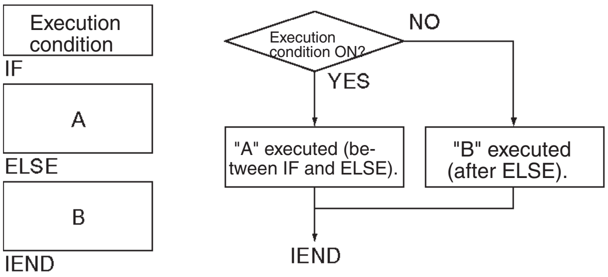
Flags

Name	Label	Operation	
		IF/IF NOT	ELSE/IEND
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.	• ON if the branch instructions are not in a block program. • OFF in all other cases.

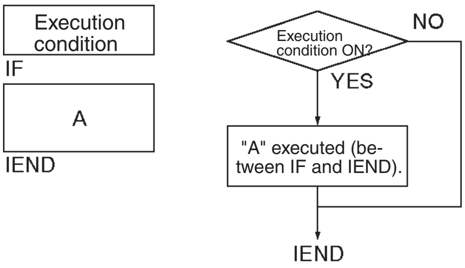
Function

● 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.

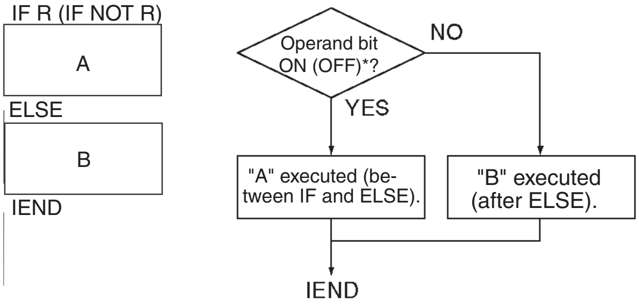


If the ELSE(803) instruction is omitted and the execution condition is ON, the instructions between IF(802) and IEND(804) will be executed and if the execution condition is OFF, only the instructions after IEND(804) will be executed.



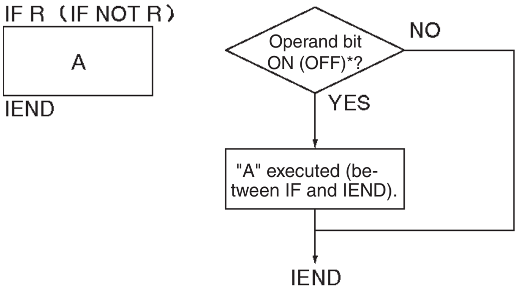
● Operation with an Operand for IF(802) or IF NOT(802)

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.

* The parentheses ( ) in the explanation indicate the case of the IF NOT instruction.

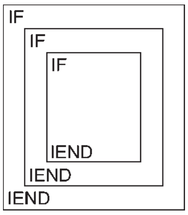


Hint

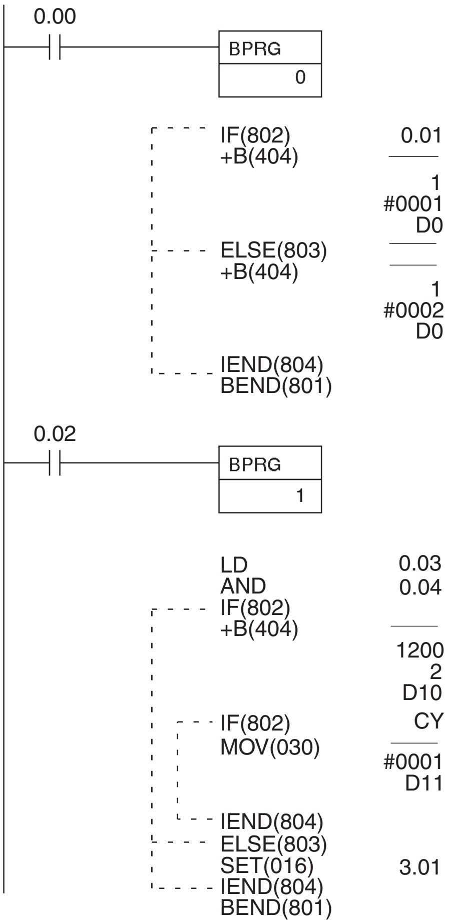
- Instructions in block programs are generally executed unconditionally. Branching, however, can be used to create conditional execution based on execution conditions or operand bits.

  Use IF A ELSE B IEND to branch between A and B.

  Use IF A IEND to branch between A and doing nothing.
- Nesting Branches
  - Up to 253 branches can be nested within the top level branch.



Example Programming



Coding

Address	Instruction	Operands
000200	LD	0.00
000201	BPRG(096)	0
000202	IF(802)	0.01
000203	+B(404)	---
		1
		#0001
		D0
000204	ELSE(803)	---
000205	+B(404)	---
		1
		#0002
		D0
000206	IEND(804)	---
000207	BEND(801)	---
000208	LD	0.02
000209	BPRG(096)	1
000210	LD	0.03
000211	AND	0.04
000212	IF(802)	---
000213	+B(404)	---
		1200
		2
		D10
000214	IF(802)	A50004
000215	MOV(030)	---
		#0001
		D11
000216	IEND(804)	---
000217	ELSE(803)	---
000218	SET(016)	3.01
000219	IEND(804)	---
000220	BEND(801)	---

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, 1 is added to the contents of CIO 1. If CIO 0.01 is OFF, 2 is added to the contents of CIO 1. 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 2 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 3.01 is turned ON.

WAIT/WAIT NOT

Instruction	Mnemonic	Variations	Function code	Function
ONE CYCLE AND WAIT	WAIT	---	805	The instructions from the WAIT instruction to the BEND instruction are not executed until the execution condition (when an operand is not specified) or the specified bit (when an operand is specified) turns ON.
ONE CYCLE AND WAIT NOT	WAIT NOT	---	805	Stops execution of the rest of the block program until a specified bit OFF or an operand bit turns OFF

	Operation without an operand (operation by execution condition)	Operation with an operand		
Symbol	Execution condition WAIT(805)	WAIT(805)	B	B: Bit operand
		WAIT(805) NOT	B	B: Bit operand

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK[*1]	OK[*1]	Not allowed

*1 WAIT(805)/WAIT(805) NOT must be used in block programming regions even within subroutines and interrupt tasks.

Operands

Operand	Description	Data type	Size
B	Bit operand	BOOL	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	OK	OK	OK[*1]	OK[*1]	---	---	---	---	---	OK	OK	OK	OK	---

*1 CJ2 CPU Units only.

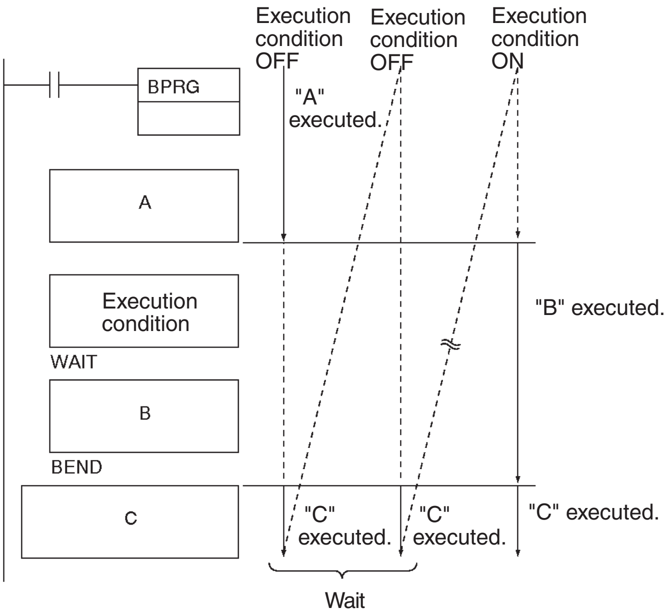
Flags

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.

Function

● Operation without an Operand

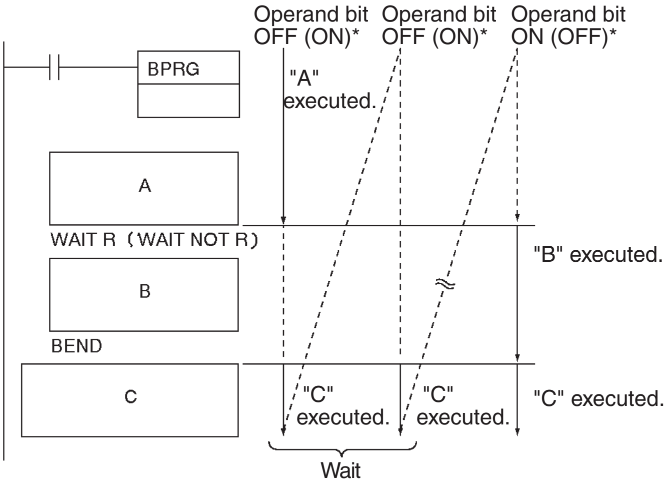
If an operand bit is not specified, an execution must be created before WAIT(805)/WAIT(805 NOT starting with LD. If the execution condition is OFF for WAIT(805), the rest of the instruction in the block program will be skipped. In the next cycle, none of the block program will be executed except for 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.

* The parentheses ( ) in the explanation indicate the case of the WAIT NOT instruction.



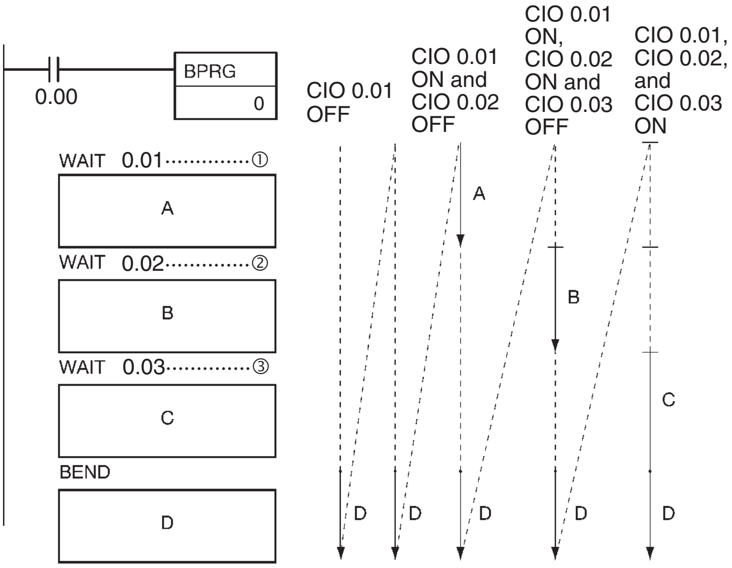
Hint

- WAIT(805) and WAIT(805) NOT can be used for step progressions inside block programs.

Precautions

- No block programming instructions will be executed while the input condition for WAIT(805) is OFF. The other block programming instructions will be executed again after the input condition for WAIT(805) turns ON. If, however, online editing is executed for a task containing a block program, the wait status created by WAIT(805) will be cleared and the block program will be executed again from the beginning.

Example Programming

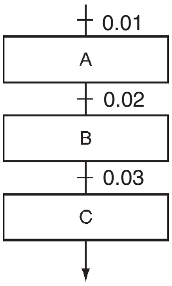


When CIO 0.00 is ON in the following example, block program 00 will be executed. Execution would proceed as follows:

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.

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.



TIMW/TIMWX

Instruction	Mnemonic	Variations	Function code	Function
HUNDRED-MS TIMER WAIT	TIMW	---	813	Delays execution of the rest of the block program until the specified time has elapsed.
	TIMWX	---	816	

	BCD	Binary
Symbol	TIMW(813) N SV N: Timer number SV: Set value	TIMWX(816) N SV N: Timer number SV: Set value

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	Not allowed	Not allowed

*1 TIMW(813)/TIMWX(816) must be used in block programming regions even within subroutines.

Operands

Operand	Description	Data type	Size
N	Timer Number	TIMER	1
SV	Set Value	WORD	1

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
SV	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

Name	Label	Operation
Error Flag	P_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.

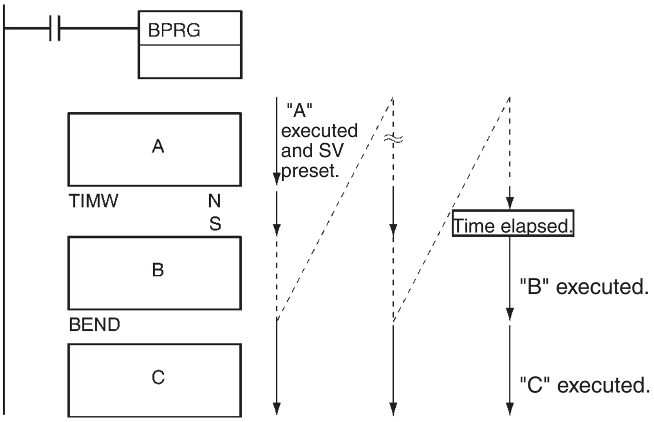
Function

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.

Note The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from –10 ms to + the cycle time.

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.



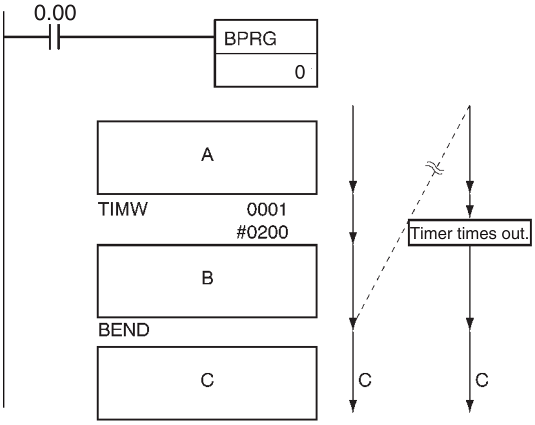
Hint

- 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.

Precautions

- No block programming instructions will be executed after the input condition for TIMW(813) turns ON until TIMW(813) times out. The other block programming instructions will be executed again after the set time for TIMW(813) has expired. If, however, online editing is executed for a task containing a block program, the wait status created by TIMW(813) will be cleared and the block program will be executed again from the beginning.
- 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.
- 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 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.
- The timer will not operate correctly if the cycle time is 100 ms or longer.

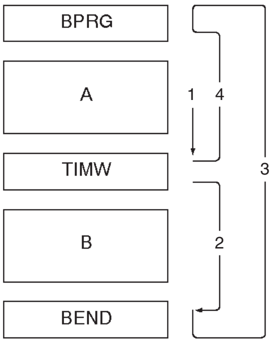
Example Programming



Coding

Address	Instruction	Operand
000200	LD	000000
000201	BPRG	0
. .	A	. .
000210	TIMW	1
		#0200
. .	B	. .
000220	BEND	---

- Operation of TIMW(813)
  When 0.00 turns ON, BPRG0 is executed. After program “A” is executed, the set value is preset in timer 1 and the program jumps to BEND.
  “A” is not executed in the next cycle; execution takes place from the TIMW instruction. When the time (20 seconds) elapses, program “B” is executed.
  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.



CNTW/CNTWX

Instruction	Mnemonic	Variations	Function code	Function
COUNTER WAIT	CNTW	---	814	Delays execution of the rest of the block program until the specified count has been achieved.
	CNTWX	---	818	

Symbol	BCD			Binary		
	CNTW(814)	N	N: Counter number	CNTWX(818)	N	N: Counter number
		SV	SV: Set value		SV	SV: Set value
		I	I: Count input		I	I: Count input

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	OK*1	Not allowed

*1 CNTW(814)/CNTWX(818) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

Operand	Description	Data type	Size
N	Counter Number	COUNTER	1
S	Set Value	WORD	1
I	Count input	BOOL	1

N: Timer Number

BCD: 0 to 4095 (decimal)

Binary: 0 to 4095 (decimal)

SV: Set Value

BCD: #0000 to #9999 (BCD)

Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	---	OK	---	---	---	---	---	---	---	OK	---	---	---	---
S	OK	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK						
I							OK*1	OK*1	---	---	---	---		OK	OK	OK	OK	

*1 CJ2 CPU Units only.

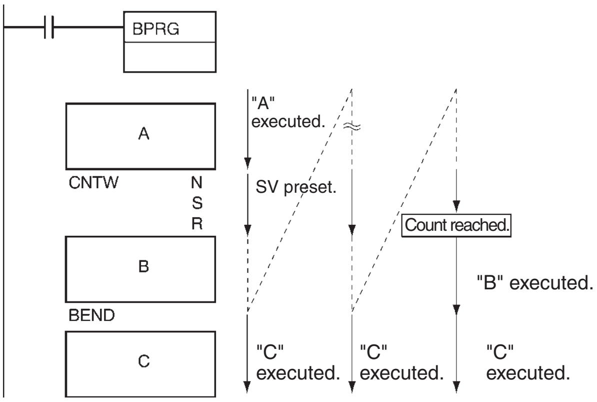
Flags

Name	Label	Operation
Error Flag	P_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.

Function

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.



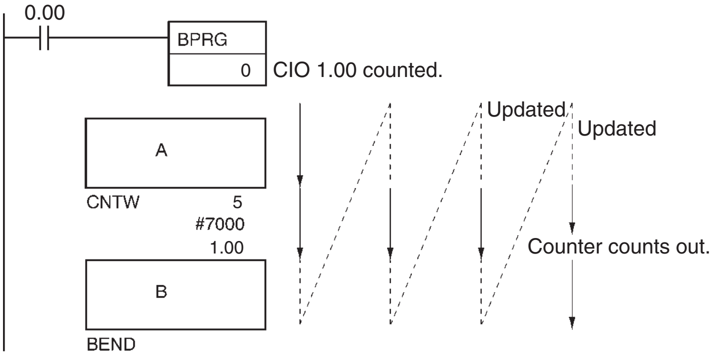
Hint

- 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.

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.

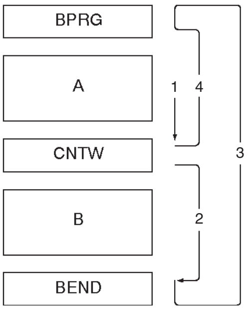
Example Programming



Coding

Address	Instruction	Operand
000200	LD	0.00
000201	BPRG	0
. .	A	. .
000210	CNTW	5
		#7000
		1.00
. .	B	. .
000220	BEND	---

- 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.
  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.



TMHW/TMHWX

Instruction	Mnemonic	Variations	Function code	Function
TEN-MS TIMER WAIT	TMHW	---	815	Delays execution of the rest of the block program until the specified time has elapsed.
	TMHWX	---	817	

	BCD			Binary		
Symbol	TMHW(815)	N SV	N: Timer number SV: Set value	TMHWX(817)	N SV	N: Timer number SV: Set value

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	Not allowed	Not allowed

*1 TMHW(815)/TMHWX(817) must be used in block programming regions even within subroutines.

Operands

Operand	Description	Data type	Size
N	Timer Number	TIMER	1
SV	Set Value	WORD	1

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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	---	---	---	---	OK	---	---	---	---	---	---	---	---	OK	---	---	---	---
SV	OK	OK	OK	OK		OK	OK	OK	OK	OK	OK	OK						

Flags

Name	Label	Operation
Error Flag	P_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.

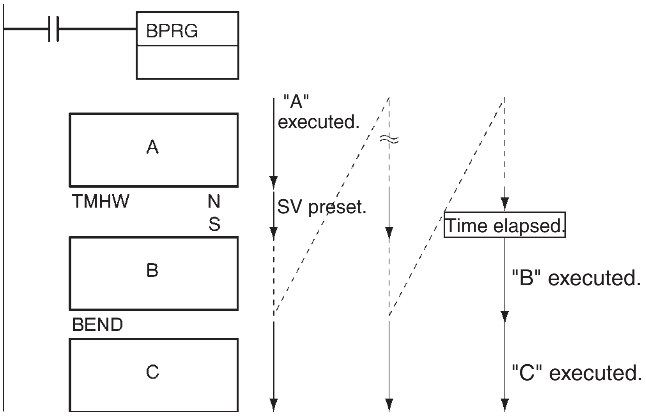
Function

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.

Note The timer accuracy for a CS1D CPU Unit for a Duplex CPU System is from −10 ms to + the cycle time.

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.



Hint

- This instruction is a WAIT instruction that uses a timer (time out) for the execution condition. This can be used in a process that advances based on a timer.

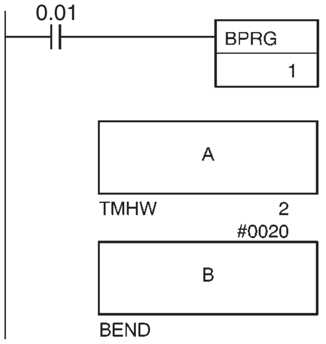
Precautions

- 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.
- 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 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.
- The timer will not operate correctly if the cycle time is 100 ms or longer.

Example Programming

- Operation of TMHW(815)

When 0.01 is ON, BPRG1 is executed. After program A is executed, the set value is preset in timer 2 and the program jumps to BEND. A is not executed in the next cycle; execution takes place from the TMHW instruction. When the time elapses (0.2 seconds), program B is executed.



Coding

Address	Instruction	Operand
000221	LD	0.01
000222	BPRG	0
. .	A	. .
000250	TMHW	2
		#0020
. .	B	. .
000281	BEND	---

LOOP/LEND/LEND NOT

Instruction	Mnemonic	Variations	Function code	Function
Loop Control	LOOP	---	809	Indicates the beginning of the loop (LOOP to LEND).
	LEND	---	810	Indicates the end of the loop (LOOP to LEND). LOOP to LEND is repeated until the execution condition (when an operand is not specified) or the specified bit (when an operand is specified) turns ON.
	LEND NOT	---	810	Indicates the end of the loop (LOOP to LEND NOT). LOOP to LEND NOT is repeated until the specified bit turns OFF.

	LOOP	LEND (LEND NOT)
Symbol	LOOP(809)	Operation without an operand (operation by execution condition) Execution condition LEND(810) Operation with an operand LEND(810) B B: Bit operand LEND(810) NOT B B: Bit operand

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	Not allowed	OK	OK	OK*1	OK*1	Not allowed

*1 LOOP(809), LEND(810), and LEND(810) NOT must be used in block programming regions even within subroutines and interrupt tasks.

Operands

Operand	Description	Data type	Size
B	Bit operand	BOOL	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
B	OK	OK	OK	OK	OK	OK	OK*1	OK*1	---	---	---	---	---	OK	OK	OK	OK	---

*1 CJ2 CPU Units only.

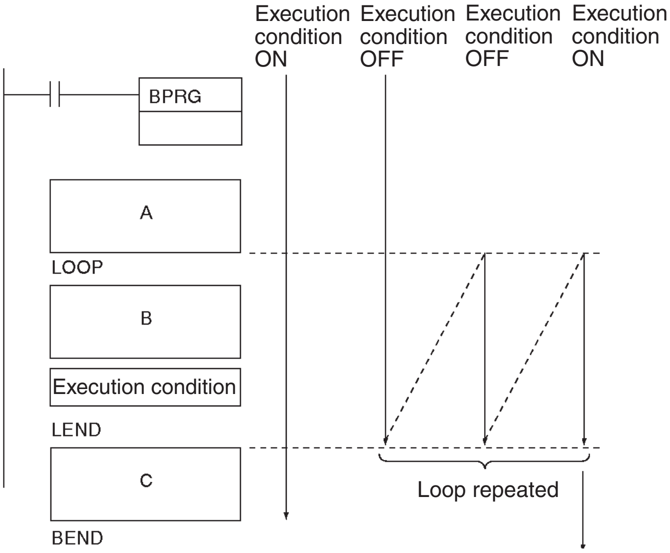
Flags

Name	Label	Operation
Error Flag	ER	• ON if a Loop Control Instruction is not in a block program. • OFF in all other cases.

Function

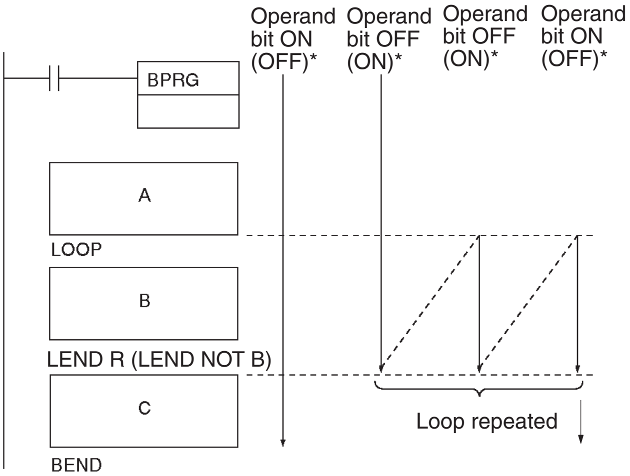
● Using an Execution Condition for LEND(810)

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.



* The status of the operand bit would be reversed for LEND(810) NOT.

Precautions

Execution inside a loop does not refresh I/O data. If I/O data must be refreshed during the loop, use IORF(184).

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.

- 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.

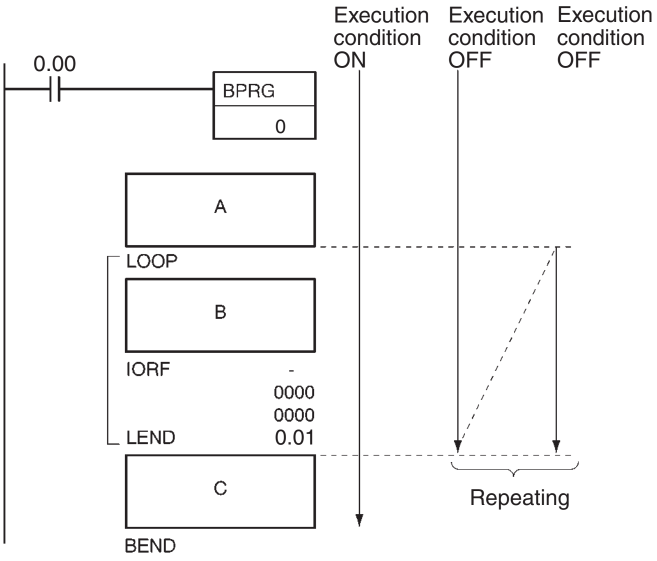
Correct:	Incorrect:
LOOP(809)	LOOP(809)
IF(802)	IF(802)
IF(802)	IF(802)
IEND(804)	IEND(804)
IEND(804)	LEND(810)
LEND(810)	IEND(804)

- NOP processing will be performed if LOOP(809) is not executed.

Example Programming

- Operation of LOOP(809)

  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.



Coding

Address	Instruction	Operand
000220	LD	0.00
000201	BPRG	0
. .	A	. .
000210	LOOP	2
. .	B	. .
000220	IORF	. .
		0000
		0000
000221	LEND	0.01
. .	C	. .
000220	BEND	---

Text String Processing Instructions

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

A	B
C	D
E	NUL

=

41	42
43	44
45	00

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

A	B
C	D
NUL	NUL

=

41	42
43	44
00	00

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.

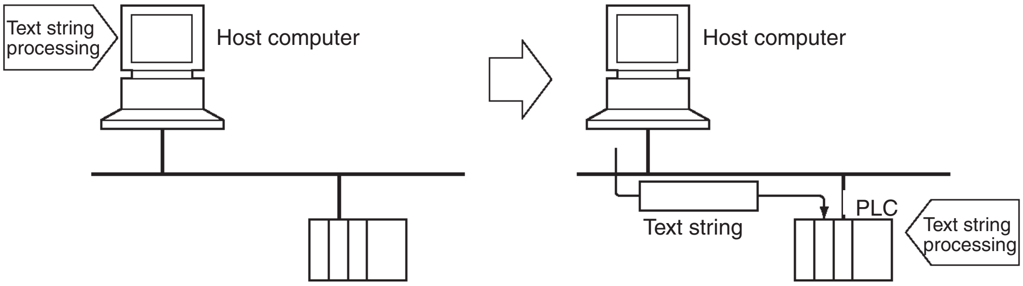
Example: MOV$ D0 D100

D0	41	42
D1	43	44
D2	45	NUL

→

D100	41	42
D101	43	44
D102	45	NUL

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.



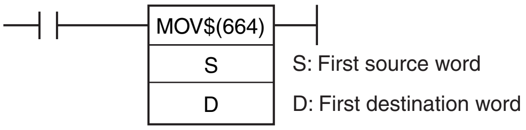
For example, production plan data such as product names can be transferred from the host computer to the PLC. Various operations such as inserting and rearranging text strings can be then be performed at the PLC, thereby reducing the data processing load at the host computer.

The ASCII characters that can be handled by text string processing instructions are shown in the following table.

	Four leftmost bits															
Four rightmost 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	゜		

MOV$

Instruction	Mnemonic	Variations	Function code	Function
MOV STRING	MOV$	@MOV$	664	Transfers a text string.

	MOV$
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	First source word	UINT	Variable
D	First destination word	UINT	Variable

● Operand Specifications

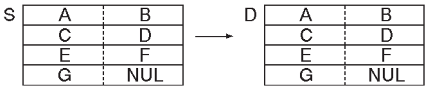
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D																		

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.
Equal Flag	=	• ON if 0000 (hex) is transferred to D. • OFF in all other cases.

Function

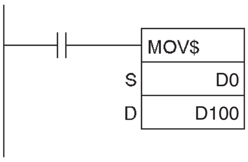
MOV$(664) transfers the text string data designated by S, just as it is, as text string data (including the final NUL), to D. The maximum number of characters that can be designated by S is 4,095 (0FFF hex).

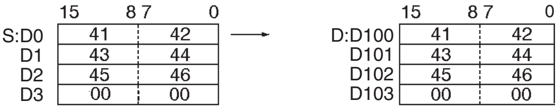


MOV$(664) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

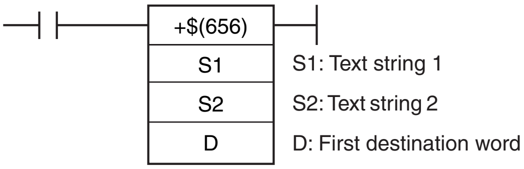
In this example, MOV$(664) is used to transfer the text string ABCDEF.





+$

Instruction	Mnemonic	Variations	Function code	Function
CONCATENATE STRING	+$	@+$	656	Links one text string to another text string.

	+$
Symbol	

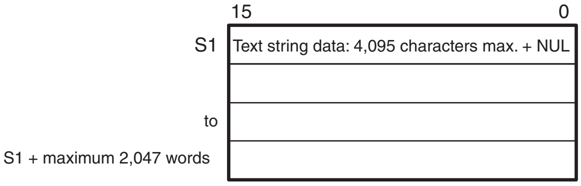
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

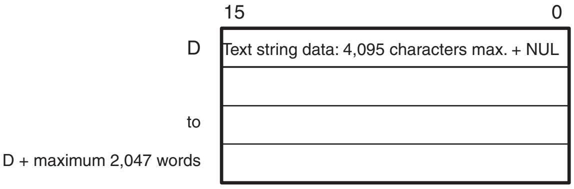
Operands

Operand	Description	Data type	Size
S1	Text string 1	INT	Variable
S2	Text string 2	INT	Variable
D	First destination word	INT	Variable

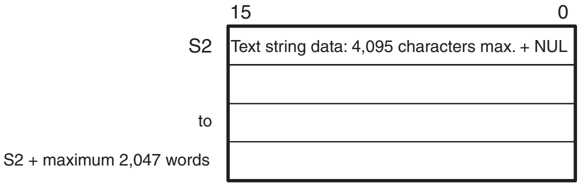
S1: Text String 1



D: First Destination Word



S2: Text String 2



Note
- 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.
- 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

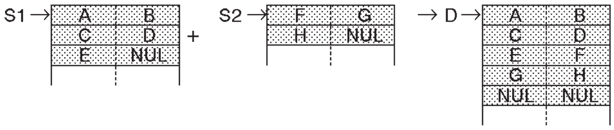
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S2																		
D																		

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.
Equal Flag	=	• ON if 0000 (hex) is transferred to D. • OFF in all other cases.

Function

+$(656) 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.

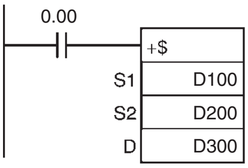
+$(656) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

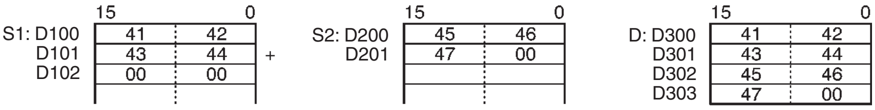
Precaution

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 Programming

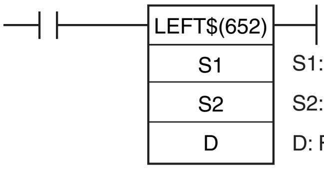
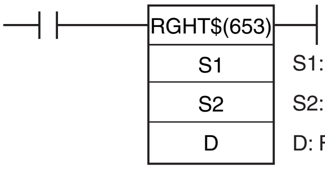
In this example, +$(656) is used to connect the text strings ABCD and EFG and output the result to D.





LEFT$/RGHT$

Instruction	Mnemonic	Variations	Function code	Function
GET STRING LEFT	LEFT$	@LEFT$	652	Fetches a designated number of characters from the left (beginning) of a text string.
GET STRING RIGHT	RGHT$	@RGHT$	653	Reads a designated number of characters from the right (end) of a text string.

	LEFT$	RGHT$
Symbol	 S1: Text string first word S2: Number of characters D: First destination word	 S1: Text string first word S2: Number of characters D: First destination word

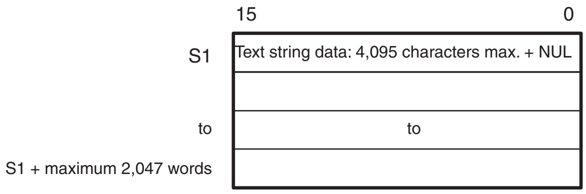
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

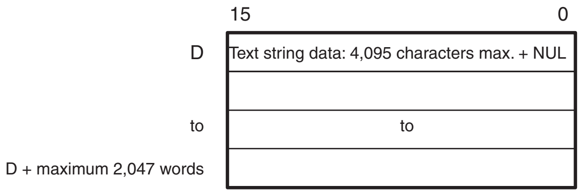
Operands

Operand	Description	Data type	Size
S1	Text string first word	UINT	Variable
S2	Number of characters	UINT	1
D	First destination word	UINT	Variable

S1: Text String



D: First Destination Word



S2: Number of Characters

0000 to 0FFF hex or &0 to &4095

Note

- 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.
- 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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S2											OK	OK						
D											---	---						

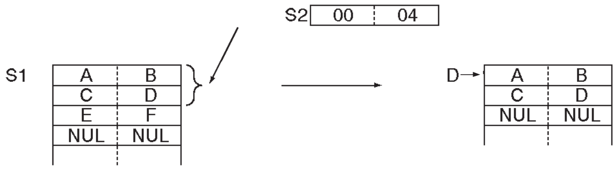
Flags

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.
Equal Flag	=	• ON if 0000 (hex) is output to D. • OFF in all other cases.

Function

● LEFT$

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).



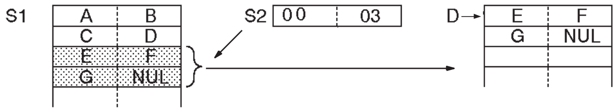
If the number of characters fetched exceeds the number of characters designated by S1, the entire S1 text string will be output.

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.

LEFT$(652) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

● RGHT$

RGHT$(653) reads the number of characters designated by S2, from the right (end) 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.

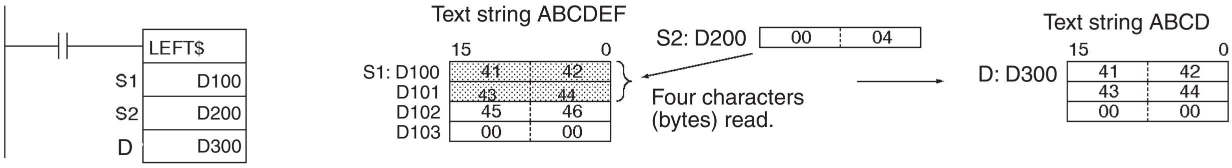
If 0 (0000 hex) is designated as the number of characters to be read, the two NUL characters (0000 hex) will be output to D.

RGHT$(653) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

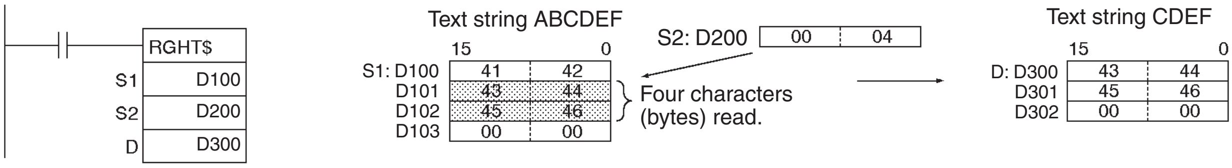
● LEFT$

In this example, LEFT$(652) is used to read four characters.



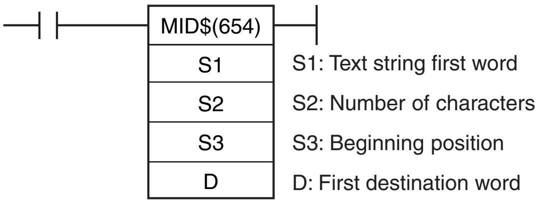
● RGHT$

In this example, RGHT$(653) is used to read four characters.



MID$

Instruction	Mnemonic	Variations	Function code	Function
GET STRING MIDDLE	MID$	@MID$	654	Reads a designated number of characters from any position in the middle of a text string.

	MID$
Symbol	

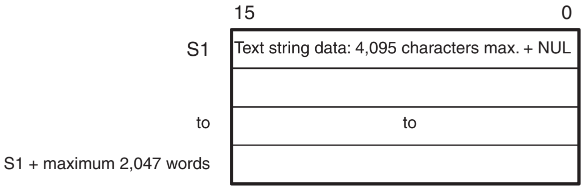
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

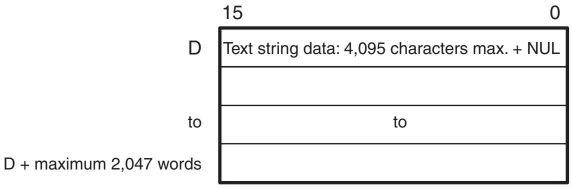
Operands

Operand	Description	Data type	Size
S1	Text string first word	UINT	Variable
S2	Number of characters	UINT	1
S3	Beginning position	UINT	1
D	First destination word	UINT	Variable

S1: Text String



D: First Destination Word



S2: Number of Characters

0000 to 0FFF hex or &0 to &4095

S3: Beginning Position

0001 to 0FFF hex or &1 to &4095

Note
- 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.
- 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	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S2, S3											OK	OK						
D											---	---						

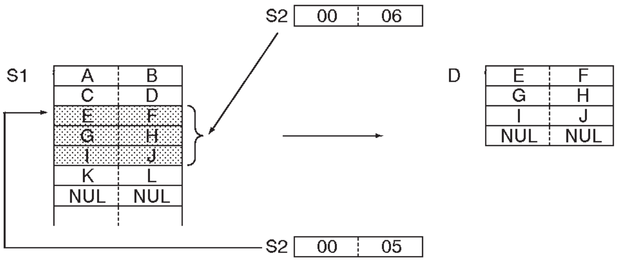
Flags

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 S3 data is not 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.
Equal Flag	=	• ON if 0000 (hex) is output to D. • OFF in all other cases.

Note If the beginning position designated by S3 is beyond the text string designated by S1, an error will be generated and the Error Flag will turn ON.

Function

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).



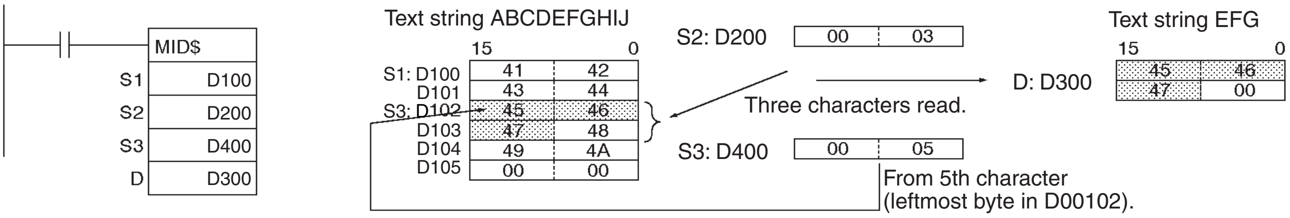
If the number of characters to be read extends beyond the end of the text string designated by S1, the string will be output up to the end.

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.

MID$(654) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

In this example, MID$(654) is used to read three characters.

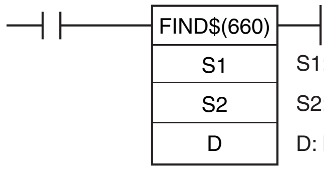


FIND$

Instruction	Mnemonic	Variations	Function code	Function
FIND IN STRING	FIND$	@FIND$	660	Finds a designated text string from within a text string.

Symbol

FIND$



S1: Source text string first word

S2: Found text string first word

D: First destination word

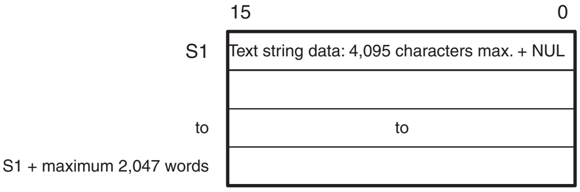
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

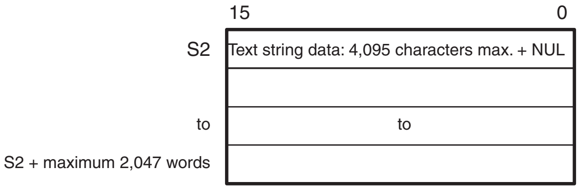
Operands

Operand	Description	Data type	Size
S1	Source text string first word	UINT	Variable
S2	Found text string first word	UINT	Variable
D	First destination word	UINT	1

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

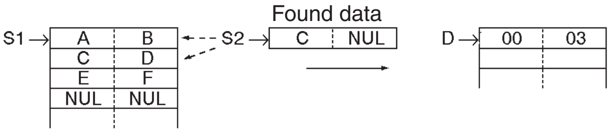
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S2																		
D																		

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.
Equal Flag	=	• ON if 0000 (hex) is output to D. • OFF in all other cases.

Function

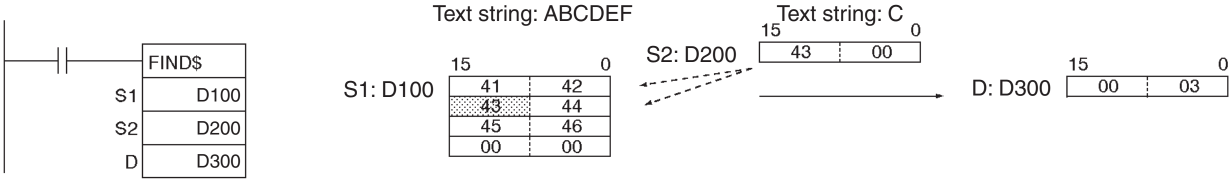
FIND$(660) finds the text string designated by S2 from within the text string designated by S1, and outputs the result (a given number of characters from the beginning of S1) in binary data to D. If there is no matching text string, 0000 hex is output to D.



FIND$(660) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

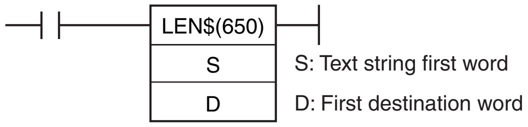
Example Programming

In this example, FIND$(660) is used to find one character from within a text string.



LEN$

Instruction	Mnemonic	Variations	Function code	Function
STRING LENGTH	LEN$	@LEN$	650	Calculates the length of a text string.

Symbol	LEN$
	

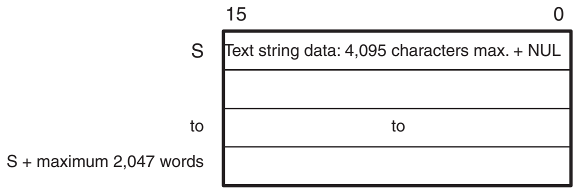
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Text string first word	UINT	Variable
D	First destination word	UINT	1

S: Text String



Note The data from S to S + the maximum 2,047 words must be in the same area.

● Operand Specifications

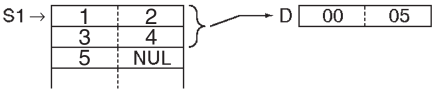
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
D												OK						

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.
Equal Flag	=	• ON if the calculated result is 0. • OFF in all other cases.

Function

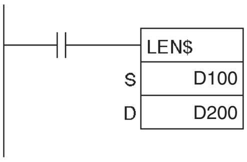
LENS$(650) calculates the number of characters from the first word of the text string, designated by S, until the NUL code (00 hex), including the NUL code itself, and outputs the result to D as binary data. If there is a NUL at the beginning of the text string, the result that is calculated will be 0000 hex.

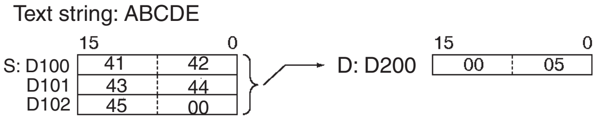


LENS$(650) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

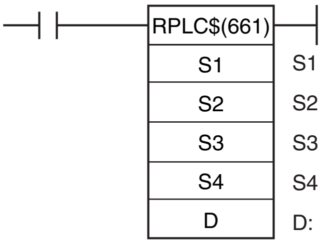
In this example, LENS$(650) is used to calculate the number of characters and output the result.





RPLC$

Instruction	Mnemonic	Variations	Function code	Function
REPLACE IN STRING	RPLC$	@RPLC$	661	Replaces a text string with a designated text string from a designated position.

	RPLC$
Symbol	 S1: Text string first word S2: Replacement text string first word S3: Number of characters S4: Beginning position D: First destination word

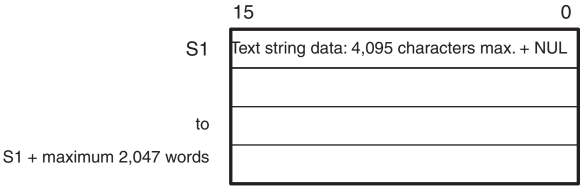
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

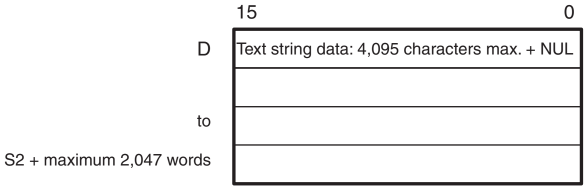
Operands

Operand	Description	Data type	Size
S1	Text string first word	UINT	Variable
S2	Replacement text string first word	UINT	Variable
S3	Number of characters	UINT	1
S4	Beginning position	UINT	1
D	First destination word	UINT	Variable

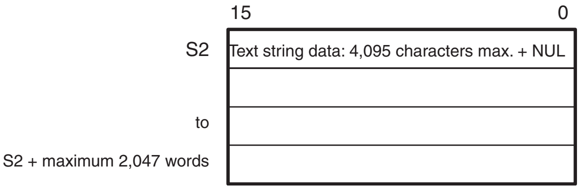
S1: Text String



D: First Destination Word



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
- 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.
- 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 cannot overlap.

● Operand Specifications

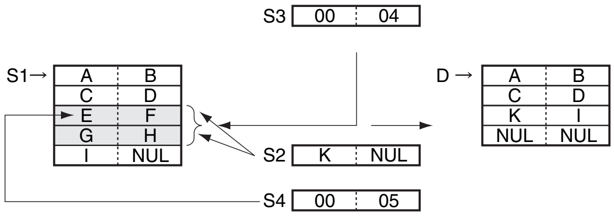
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1, S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S3, S4	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
D	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

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 not 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.
Equal Flag	=	• ON if 0000 (hex) is output to D. • OFF in all other cases.

Function

PLC$(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.



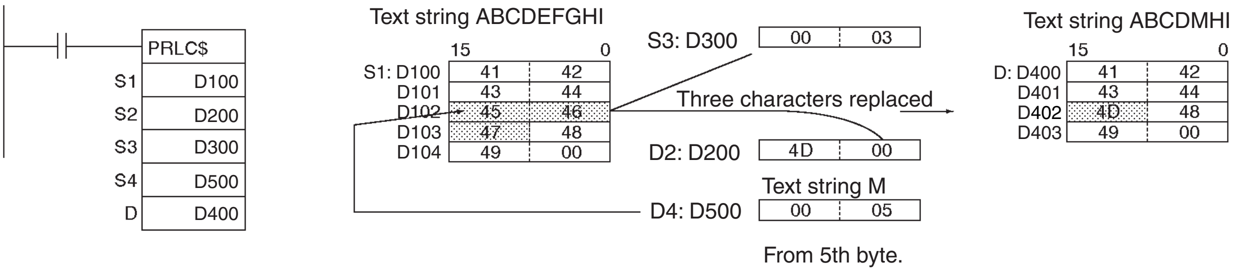
- The maximum number of characters in the result is 4,095 (0FFF hex). If the number is greater than that, only 4,095 characters will be output (with NUL added as the 4,096th).
- From 0 to 4,095 characters (0000 to 0FFF hex) can be replaced. If the number is 0, then the text string designated by S1 will 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. RPLC$(661) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Precaution

- 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.

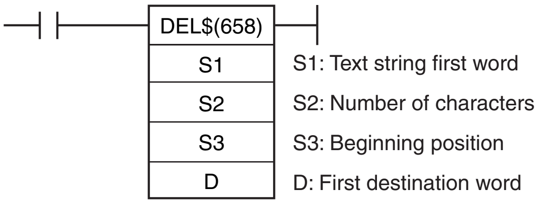
Example Programming

In this example, RPLC$(654) is used to replace three characters.



DEL$

Instruction	Mnemonic	Variations	Function code	Function
DELETE STRING	DEL$	@DEL$	658	Deletes a designated text string from the middle of a text string.

	DEL$
Symbol	

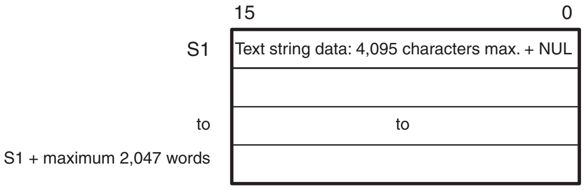
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

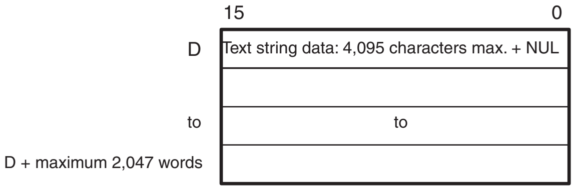
Operands

Operand	Description	Data type	Size
S1	Text string first word	UINT	Variable
S2	Number of characters	UINT	1
S3	Beginning position	UINT	1
D	First destination word	UINT	Variable

S1: Text String



D: First Destination Word



S2: Number of Characters

(0000 to 0FFF hex or &0 to &4095)

S3: Beginning Position

(0001 to 0FFF hex or &1 to &4095)

Note
- 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.
- The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

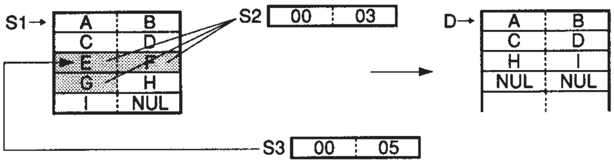
● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S2, S3											OK	OK						
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 S3 data is not 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.
Equal Flag	=	• ON when 0000 hex is output to D. • OFF in all other cases.

Function



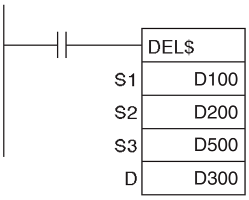
Within the text string designated by S1, DEL$(658) deletes the number of characters designated by S2, from the beginning word designated by S3, and outputs the result to D as text string data (with NUL added at the end).

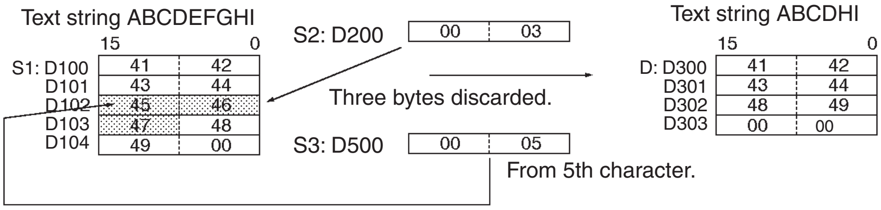
If the number of characters to be deleted extends beyond the end of the S1 text string, all of the characters up to the end will be deleted. If all of the characters from the beginning of S1 to the end are designated to be deleted, then 000 hex will be output to D.

DEL$(658) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

In this example, DEL$(658) is used to discard three characters.



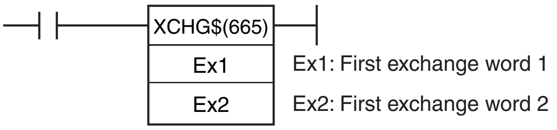


XCHG$

Instruction	Mnemonic	Variations	Function code	Function
EXCHANGE STRING	XCHG$	@XCHG$	665	Replaces a designated text string with another designated text string.

Symbol

XCHG$



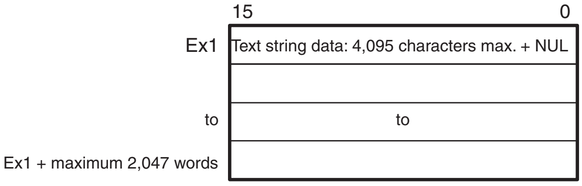
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

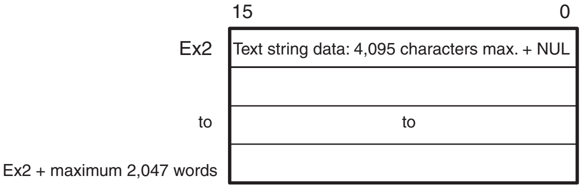
Operands

Operand	Description	Data type	Size
Ex1	First exchange word 1	UINT	Variable
Ex2	First exchange word 2	UINT	Variable

Ex1: First Exchange Word 1



Ex2: First Exchange Word 2



Note
- 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.
- 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

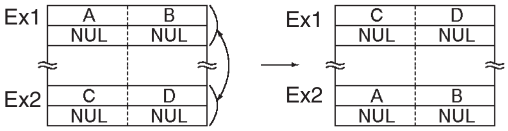
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
D1, D2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

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.

Function

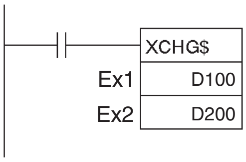
XCHG$(665) exchanges the text string designated by Ex1 with the text string designated by Ex2. If either Ex1 or Ex2 is NUL, then two NUL characters (0000 hex) will be output to the other one of them.

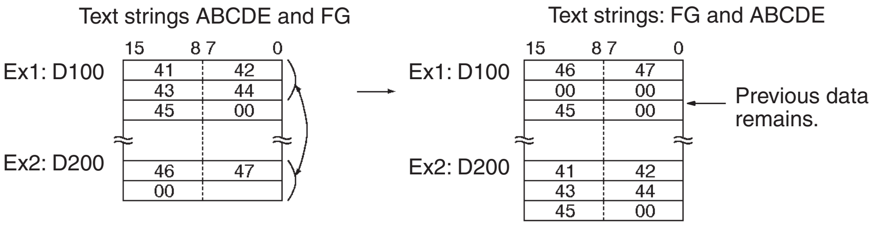


XCHG$(665) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

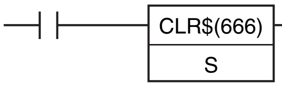
In this example, XCHG$(665) is used to exchange two text strings.





CLR$

Instruction	Mnemonic	Variations	Function code	Function
CLEAR STRING	CLR$	@CLR$	666	Clears an entire text string with NUL (00 hex).

Symbol	CLR$
	 S: Text string first word

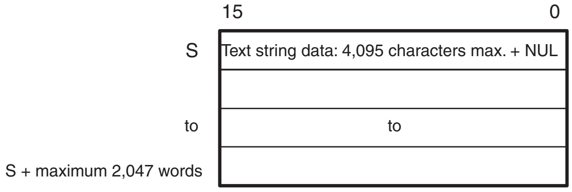
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Text string first word	UINT	Variable

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

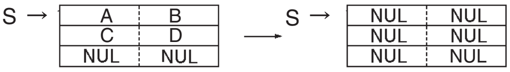
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

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.

Function

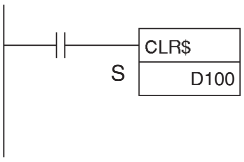
CLR$(666) clears with NUL (00 hex) the entire text string from the first word designated by S until the NUL code (00 hex). The maximum number of characters that can be cleared is 4,096. If there is no NUL before the 4,096th character, only 4,096 characters will be cleared

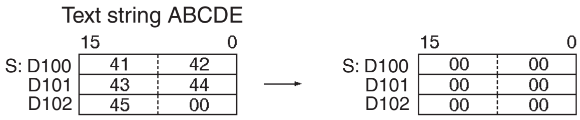


CLR$(666) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Example Programming

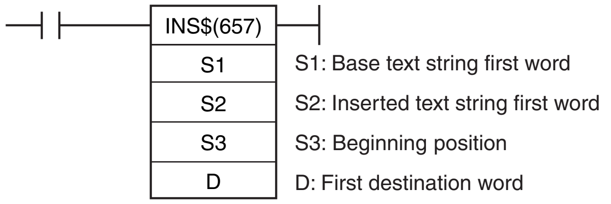
In this example, CLR$(666) is used to clear text string ABCDE.





INS$

Instruction	Mnemonic	Variations	Function code	Function
INSERT INTO STRING	INS$	@INS$	657	Inserts a designated text string into the middle of a text string.

	INS$
Symbol	

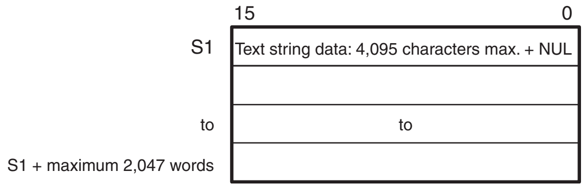
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

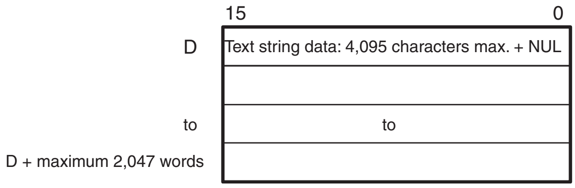
Operands

Operand	Description	Data type	Size
S1	Base text string first word	UINT	Variable
S2	Inserted text string first word	UINT	Variable
S3	Beginning position	UINT	1
D	First destination word	UINT	Variable

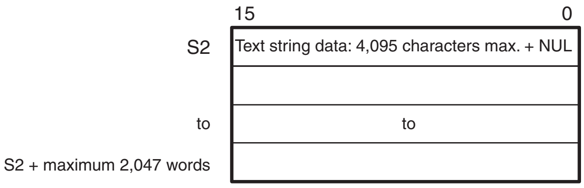
S1: Base Text String



D: First Destination Word



S2: Inserted Text String



S3: Beginning Position

(0000 to 0FFF hex or &0 to &4095)

Note
- 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.
- 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

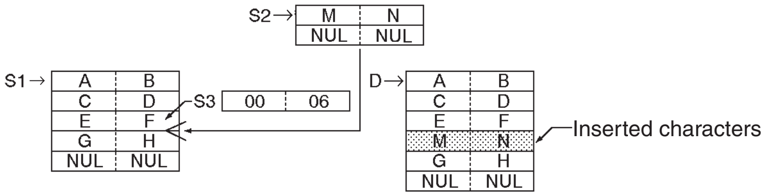
Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1, S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
S3											OK	OK						
D											---	---						

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.
Equal Flag	=	• ON if 0000 (hex) is output to D. • OFF in all other cases.

Function

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).



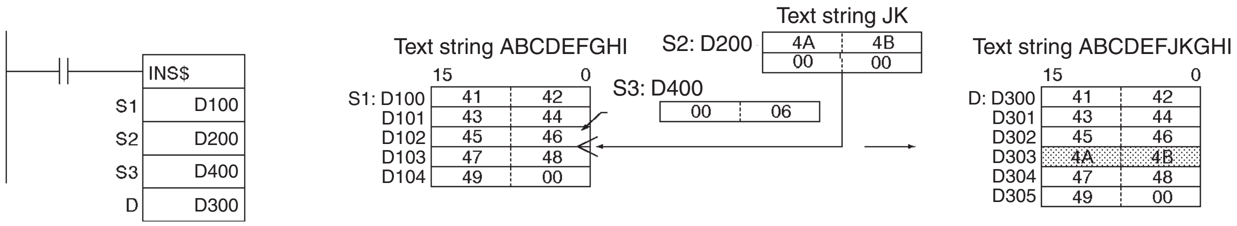
- The maximum number of characters that can be inserted is 4,095 (0FFF hex). If there are more than that, only 4,095 characters will be output to D (with NUL added as the 4,096th character).
- The 0th to the 4095th character can be specified for the starting position (S3) of the insertion. If the 0th character is specified, the operation is the same as joining S2 and S1.
- 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.
- INS$(657) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) or the *CJ2 CPU Unit Software Operation Manual* (W473) for details.

Precaution

- 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.

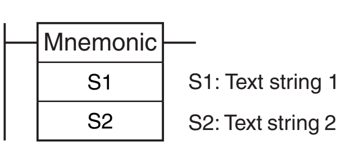
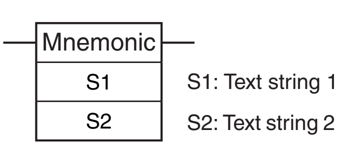
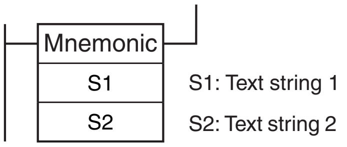
Example Programming

In this example, INS$(657) is used to insert two characters.



=$, <>$, <$, <=$, >$, >=$

Instruction	Mnemonic	Variations	Function code	Function
String Comparison	=$, <>$, <$, <=$, >$, >=$	---	670 to 675	String 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.

	LD (Load)	AND (Series Connection)	OR (Parallel Connection)
Symbol			

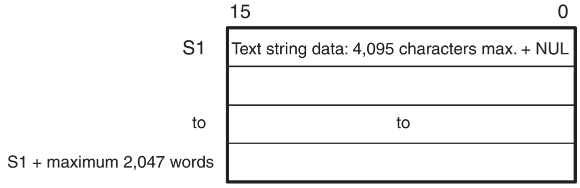
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

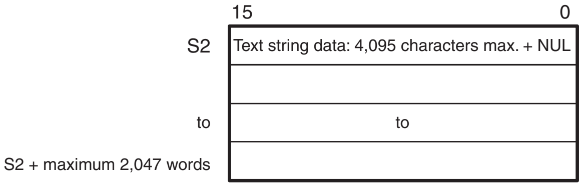
Operands

Operand	Description	Data type	Size
S1	Text string 1	UINT	Variable
S2	Text string 2	UINT	Variable

S: Text String 1



S: Text String 2



Note
- 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.
- 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	Word addresses								Indirect DM/EM addresses		Constants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S1, S2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---

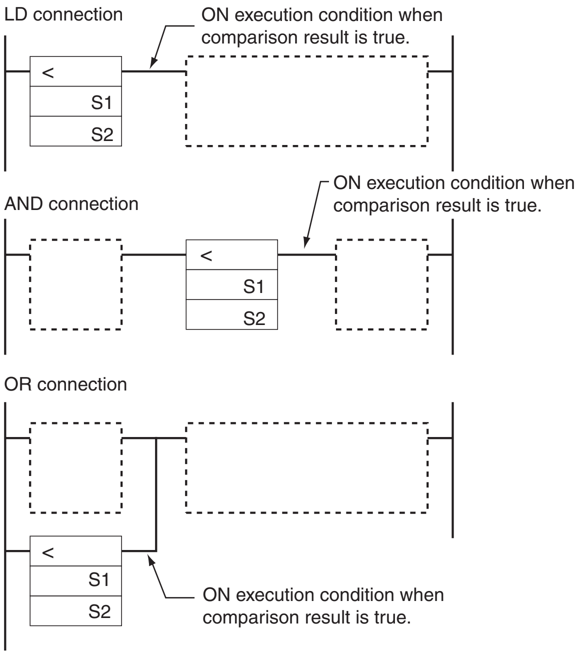
Flags

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 When the Error Flag is ON, an OFF execution condition will be output to the next instruction.

Function

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).



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.

String comparison instructions are expressed by 18 types of mnemonics.

Text string comparison instructions

The following table provides details of these instructions.

Function	Mnemonic	Name	Code
True when S1 text string equals S2 text string.	LD=$(670)	LOAD STRING EQUALS	670
	AND=$(670)	AND STRING EQUALS	
	OR=$(670)	OR STRING EQUALS	
True when S1 text string does not equal S2 text string.	LD<>$(671)	LOAD STRING NOT EQUAL	671
	AND<>$(671)	AND STRING NOT EQUAL	
	OR<>$(671)	OR STRING NOT EQUAL	
True when S1 text string is less than S2 text string.	LD<$(672)	LOAD STRING LESS THAN	672
	AND<$(672)	AND STRING LESS THAN	
	OR<$(672)	OR STRING LESS THAN	
True when S1 text string is less than or equal to S2 text string.	LD<=$(673)	LOAD STRING LESS THAN OR EQUALS	673
	AND<=$(673)	AND STRING LESS THAN OR EQUALS	
	OR<=$(673)	OR STRING LESS THAN OR EQUALS	
True when S1 text string is greater than S2 text string.	LD>$(674)	LOAD STRING GREATER THAN	674
	AND>$(674)	AND STRING GREATER THAN	
	OR>$(674)	OR STRING GREATER THAN	
True when S1 text string is greater than or equal to S2 text string.	LD>=$(675)	LOAD STRING GREATER THAN OR EQUALS	675
	AND>=$(675)	AND STRING GREATER THAN OR EQUALS	
	OR>=$(675)	OR STRING GREATER THAN OR EQUALS	

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.

Example: AD (414400 hex) and BC (424300 hex):
AD < BC, because at the beginning of the text strings 41 (hex) is less than 42 (hex).

Example: ADC (41444300 hex) and B (4200 hex):
ADC < B, because at the beginning of the text strings 41 (hex) is less than 42 (hex).

Example: 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.

Example: 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.

Example: 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.

Hint

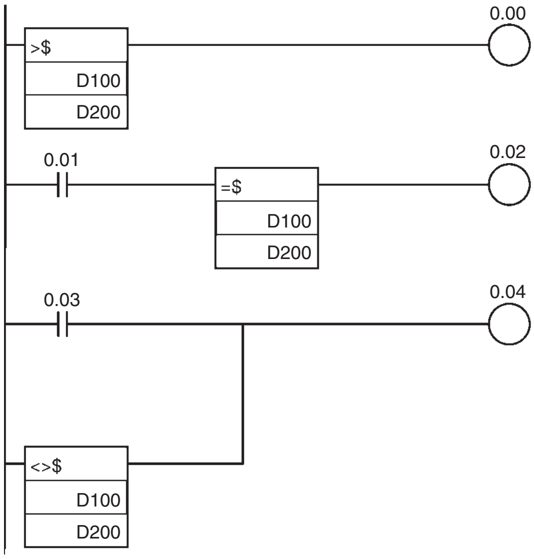
- String comparison instructions are used to rearrange the order of text strings in order of ASCII. For example, the ASCII order from lower to higher is the order of the alphabet from A to Z, so text strings can be arranged in alphabetical order.

Precaution

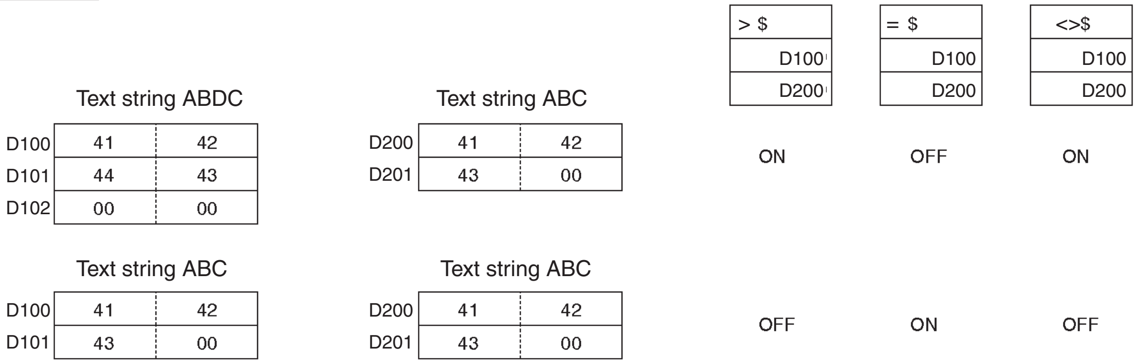
- Place a right-hand instruction after these instructions. The String Comparison Instructions cannot appear on the right side of the ladder diagram.
- These instructions cannot be used on the last rung of a logic block.

Example Programming

In this example, string comparison instructions are used to compare data.



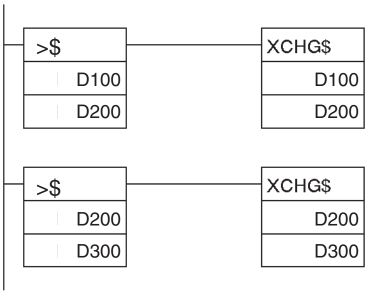
Address	Mnemonic	Operand
000000	LD > $	--- D100 D200
000001	OUT	0.00
000002	LD	0.01
000003	AND=$	--- D100 D200
000004	OUT	0.02
000005	LD	0.03
000006	OR <> $	--- D100 D200
000007	OUT	0.04



In this example, three text strings are rearranged in alphabetical order. The original order is as follows:

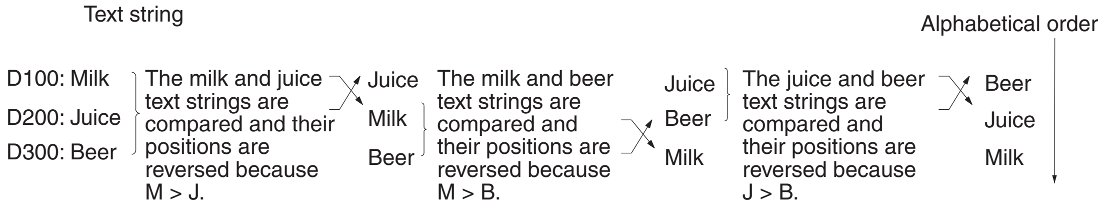
D100: Milk
D200: Juice
D300: Beer

When rearranged alphabetically, the order changes 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.



The milk and juice text strings are compared and their positions are reversed because M > J.

The milk and beer text strings are compared and their positions are reversed because M > B.

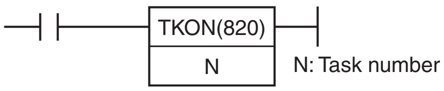
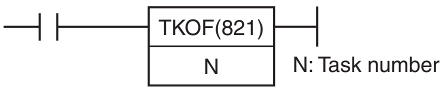
The juice and beer text strings are compared and their positions are reversed because J > B.

In this way, three text strings can be rearranged in alphabetical order.

Task Control Instructions

TKON/TKOF

Instruction	Mnemonic	Variations	Function code	Function
TASK ON	TKON	@TKON	820	Makes the specified task executable.
TASK OFF	TKOF	@TKOF	821	Puts the specified task into standby status.

	TKON	TKOF
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
N	Task number	---	1

N: Task number

- Cyclic tasks:
  N must be a constant between 0 and 127 decimal. (Values 0 to 127 specify cyclic tasks 0 to 127.)
- Extra cyclic tasks (CJ2, CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only):
  N must be a constant between 8000 and 8255 decimal. (Values 8000 to 8255 specify extra cyclic tasks 0 to 255.)

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			TK	CF	Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR				
N	---	---	---	---	---	---	---	---	---	---	OK	---	---	---	---	---	---	---

Flags

Name	Label	Operation
Error Flag	ER	• ON if N is not a constant between 00 and 127 or between 8000 and 8255 (CJ2, CS1-H, CJ1-H, and CJ1M CPU Units only). • ON if the task specified with N does not exist. • ON if TKON(820)/TKOF(821) is executed in an interrupt task. • OFF in all other cases.

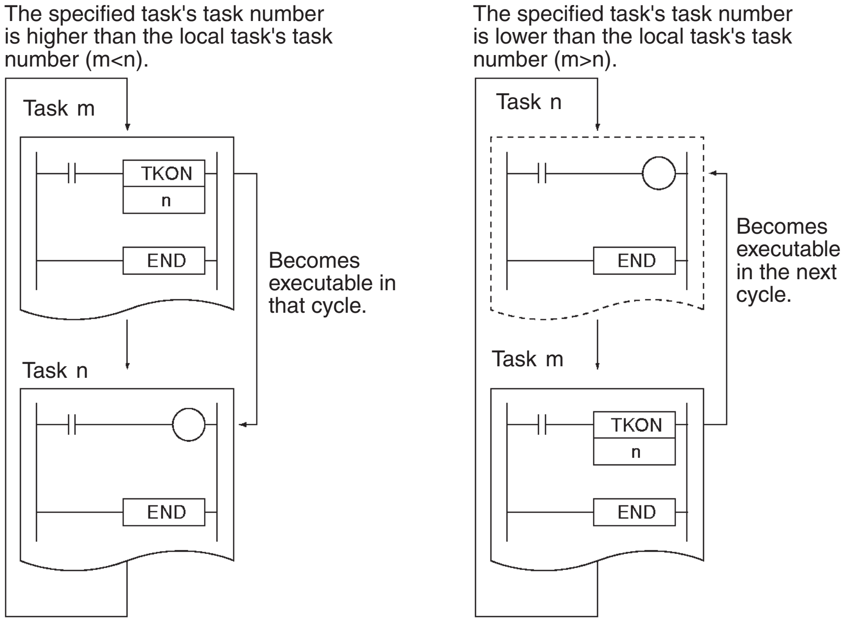
Related Auxiliary Area Words and Bits

Name	Address	Contents
Task Flags	TK00 to TK127	• 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 TK127 correspond to cyclic task numbers 00 to 127.

Function

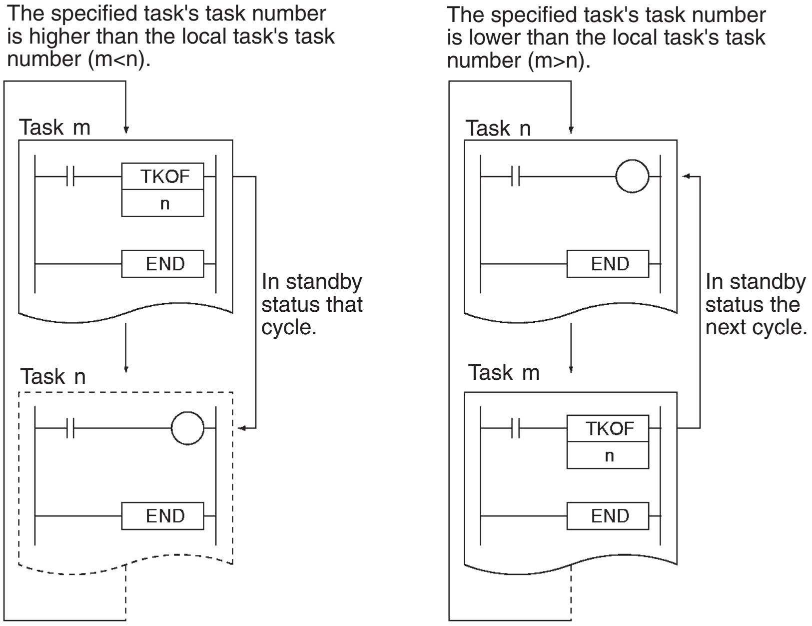
● TKON

- TKON(820) puts the specified cyclic task or extra cyclic task in executable status. When N is 0 to 127 (specifying a cyclic task), the corresponding Task Flag (TK00 to TK127) 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 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.



● TKOF

- TKOF(821) puts the specified cyclic task or extra cyclic into standby status and turns OFF the corresponding Task Flag (TK00 to TK127).
  WAIT status means that the status will not change to RUN (the program will not be executed) during that cycle.
- 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 task specified in TKOF(821) will be also in standby status in later cycles as long as it is not put into executable status by TKON(820), a Peripheral Device running 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.



Hint

- 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.
  (If the memory all clear operation is executed from the Programming Console, however, cyclic task 0 will automatically be made executable.)
- If a task is in non-executable status, TKON(820) can be 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.

● TKON

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.

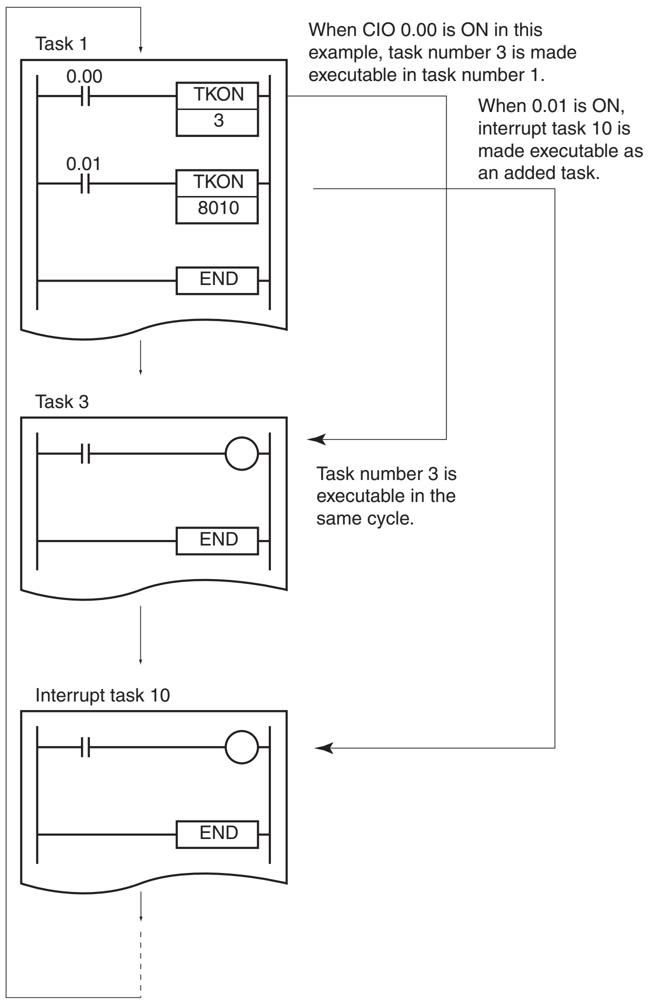
● TKOF

Cyclic tasks or extra cyclic tasks that are in executable status can be put into standby status by the TKOF(821) instruction.

Example Programming

● TKON

(Example 1)

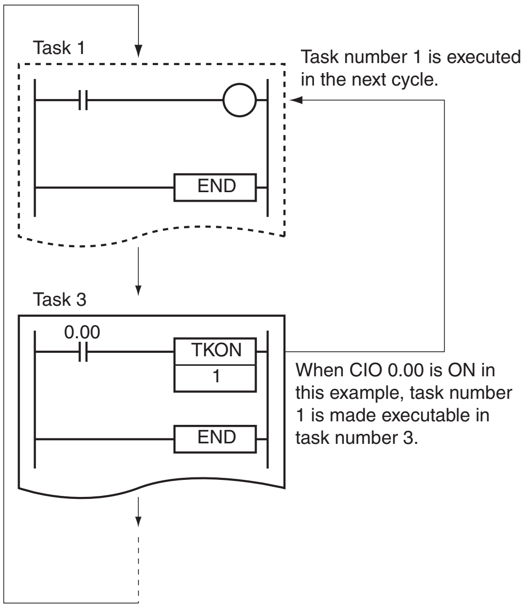


When CIO 0.00 is ON in this example, task number 3 is made executable in task number 1.

When 0.01 is ON, interrupt task 10 is made executable as an added task.

Task number 3 is executable in the same cycle.

(Example 2)

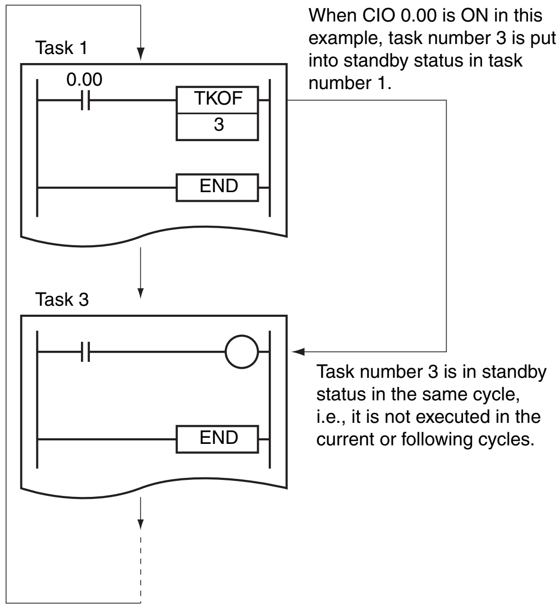


Task number 1 is executed in the next cycle.

When CIO 0.00 is ON in this example, task number 1 is made executable in task number 3.

● TKOF

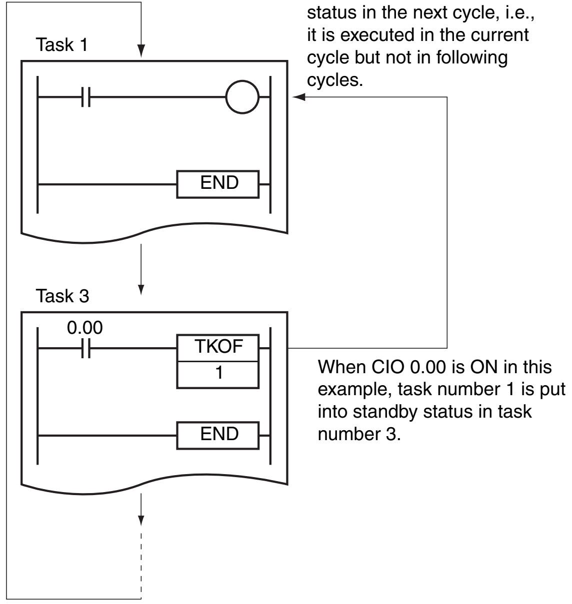
(Example 1)



When CIO 0.00 is ON in this example, task number 3 is put into standby status in task number 1.

Task number 3 is in standby status in the same cycle, i.e., it is not executed in the current or following cycles.

(Example 2)



Task number 1 is in standby status in the next cycle, i.e., it is executed in the current cycle but not in following cycles.

When CIO 0.00 is ON in this example, task number 1 is put into standby status in task number 3.

Model Conversion Instructions

Model Conversion Instructions

This section describes instructions used when changing PLC models.

- BLOCK TRANSFER (XFERC)
- SINGLE WORD DISTRIBUTE (DISTC)
- DATA COLLECT (COLLC)
- MOVE BIT (MOVBC)
- BIT COUNTER (BCNTC)

The model conversion instructions provide the same functionality as other instructions but use BCD data for the operands, like C-series instructions. (The CJ/CS-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 CS/CJ-series programs without changing the operand data for these instructions.

When converting C-series programs to CS/CJ-series programs on CX-Programmer version 5.0 higher (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.

When converting C-series programs to CS/CJ-series programs on CX-Programmer version 4.0 or lower (see note), any operand for which a constant is specified will be converted from BCD to binary, but any operand data for which a word address is specified will have to be corrected manually.

Note Conversion is achieved by specifying the CS/CJ 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 (CS1-H CPU Units, CJ1-H CPU Units, CJ1M CPU Units with unit version 3.0 or later only)	Corresponding C-series instruction	Differences from C-series instructions		When converting device type to CS/CJ with CX-Programmer Ver. 4.0 or lower	When converting device type to CS/CJ with CX-Programmer Ver. 5.0 or higher
	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	Converted to XFER. If a word address is specified for the first operand (number of words to transfer), it will need to be corrected manually to binary data in the program.	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)	Converted to DIST. If a word address is specified for the third operand (offset data), it will need to be corrected manually to binary data in the program.	DIST is converted to DICTC. Operands do not require correction.

Name	Model conversion instruction (CS1-H CPU Units, CJ1-H CPU Units, CJ1M CPU Units with unit version 3.0 or later only)	Corresponding C-series instruction	Differences from C-series instructions		When converting device type to CS/CJ with CX-Programmer Ver. 4.0 or lower	When converting device type to CS/CJ with CX-Programmer Ver. 5.0 or higher
	Mnemonic (function code)	Mnemonic (function code)	C200H, C1000H, or C2000H	C200HS, C2000HX/HG/HE(-Z), CQM1, CQM1H, CPM1/CPM1A, CPM2C, or SRM1		
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)	Converted to COLL. If a word address is specified for the second operand (offset data), it will need to be corrected manually to binary data in the program.	COLL is converted to COLLC. Operands do not require correction.
MOVE BIT	MOVBC(568)	MOVB(82)	Same	Same	Converted to MOVB. If a word address is specified for the second operand (control data), it will need to be corrected manually to binary data in the program.	MOVB is converted to MOVBC. Operands do not require correction.
BIT COUNTER	BCNTC(621)	BCNT(67)	Same	Same	Converted to BCNT. If a word address is specified for the first operand (number of words to count), it will need to be corrected manually to binary data in the program.	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 Previous CS/CJ-series Instructions

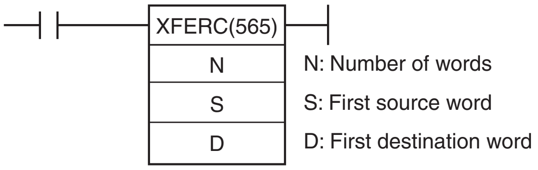
Name	Model conversion instruction (Unit Ver. 3.0 or later and CJ2 CPU Units)	Corresponding C-series instruction	Differences from previous CS/CJ-series instructions
	Mnemonic (function code)	Mnemonic (function code)	
BLOCK TRANSFER	XFERC(565)	XFER(70)	The data type for the first operand (number of words 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.

XFERC

Instruction	Mnemonic	Variations	Function code	Function
BLOCK TRANSFER	XFERC	@XFERC	565	Transfers the specified number of consecutive words.

	XFERC
Symbol	

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

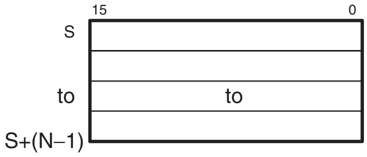
Operand	Description	Data type	Size
N	Number of words	WORD	1
S	First source word	WORD	Variable
D	First destination word	WORD	Variable

N: Number of Words

Specifies the number of words to be transferred. The possible range for N is 0000 to 9999 BCD.

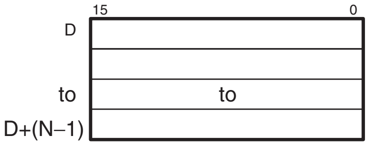
S: First Source Word

Specifies the first source word.



D: First Destination Word

Specifies the first destination word.



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
D																		

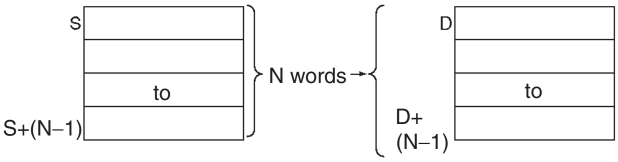
Flags

Name	Label	Operation
Error Flag	P_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.

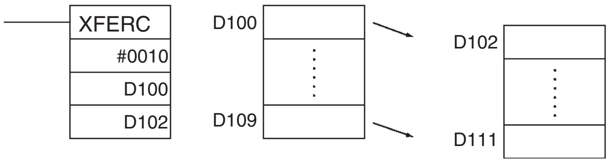
Function

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)).



Hint

- It is possible for the source words and destination words to overlap, so XFERC(565) can perform word-shift operations.

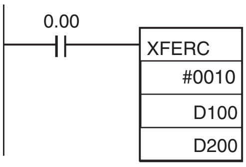


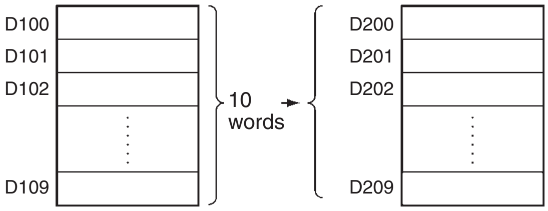
Precaution

- 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.
- 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.

Example Programming

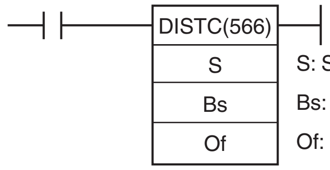
When CIO 0.00 is ON in the following example, the 10 words D100 through D109 are copied to D200 through D209.





DISTC

Instruction	Mnemonic	Variations	Function code	Function
SINGLE WORD DISTRIBUTE	DISTC	@DISTC	566	Transfers the source word to a destination word calculated by adding an offset value to the base address.

	DISTC
Symbol	 S: Source word Bs: Destination base address Of: Offset

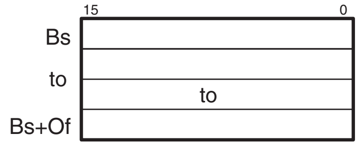
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word	WORD	1
Bs	Destination base address	WORD	1
Of	Offset	WORD	1

Bs: Destination Base Address



Of: Offset

- Data Distribution Operation (0000 to 7999 BCD)
- Stack Push Operation (9000 to 9999 BCD)

Note Bs and Bs+Of must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
Bs											---	---						
Of											OK	OK						

Flags

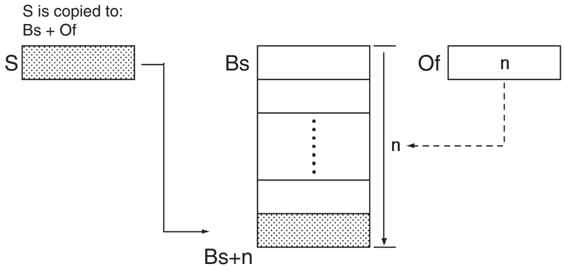
Name	Label	Operation
Error Flag	P_ER	• ON if the offset data in Of is not BCD. • ON if Stack Push Operation is specified, but the stack pointer data in Bs is not BCD. • ON if Stack Push Operation is specified and the stack pointer indicates a word that exceeds the stack data area.
Equals Flag	P_EQ	• ON if the source data is 0000. • OFF in all other cases.
Negative Flag	P_N	• ON if the leftmost bit of the source data is 1. • 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.

Function

● Data Distribution Operation

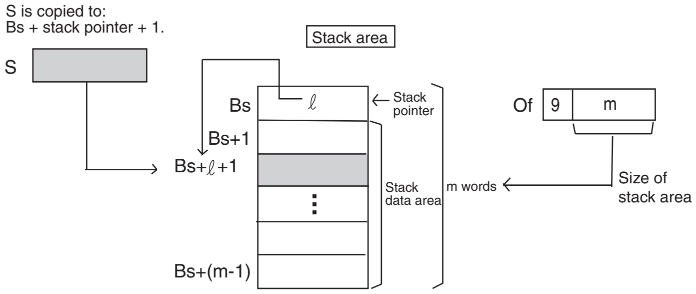
DISTC(566) copies S to the destination word calculated by adding Of to Bs.



● 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.



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 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.

Hint

- Use COLLC(567) to read stack data from the stack area.
- 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.

Precaution

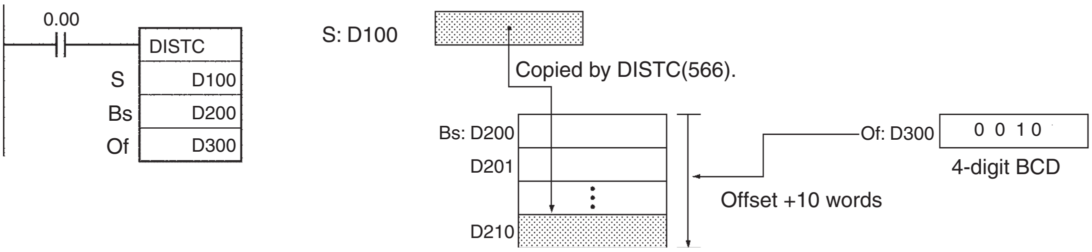
- Be sure that the offset or stack size specified by Of does not exceed the end of the data area when added to Bs.

Example Programming

● Data Distribution Operation

The leftmost byte of D300 is 0, so DISTC(566) performs the Data Distribution Operation.

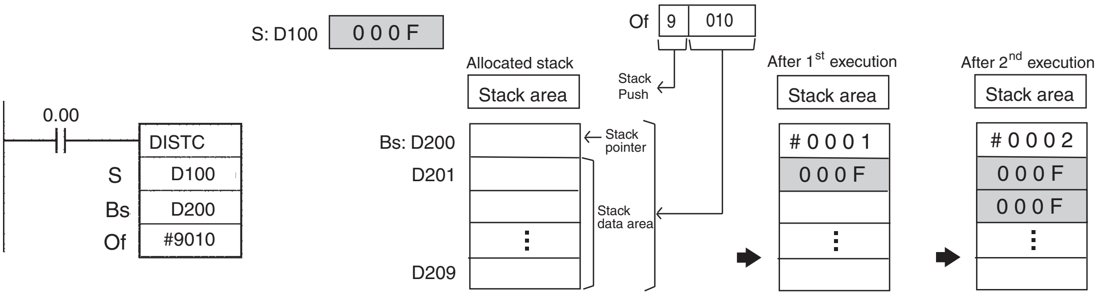
When CIO 0.00 is ON in the following example, the contents of D100 will be copied to D210 (D200 + 10) if the content of D300 is 0010 BCD. The content of D100 can be copied to other words by changing the offset in D300.



● Stack Push Operation

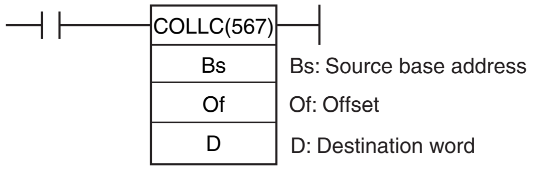
The leftmost byte of Of is 9, so DISTC(566) performs the Stack Push Operation.

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.



COLLC

Instruction	Mnemonic	Variations	Function code	Function
DATA COLLECT	COLLC	@COLLC	567	Transfers the source word (calculated by adding an offset value to the base address) to the destination word.

Symbol	COLLC
	 Bs: Source base address Of: Offset D: Destination word

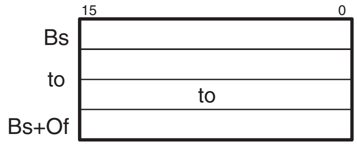
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
Bs	Source base address	WORD	1
Of	Offset	WORD	1
D	Destination word	WORD	1

Bs: Source base address



Of: Offset

- Data Collect Operation (Of = 0000 to 7999 BCD)
- FIFO Stack Read Operation (Of = 9000 to 9999 BCD)
- LIFO Stack Read Operation (Of = 8000 to 8999 BCD)

Note Bs and Bs+Of must be in the same data area.

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
Bs	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	---	---	OK	---	---	---	---
Of											OK	OK						
D											---							

Flags

Name	Label	Operation
Error Flag	P_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	P_EQ	• ON if the source data is 0000. • OFF in all other cases.
Negative Flag	P_N	• ON if the leftmost bit of the source data is 1. • OFF in all other cases.

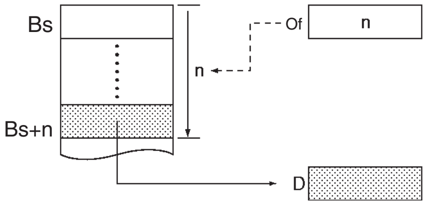
Note In C-series PLCs, the DATA COLLECT (COLL) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded.
COLLC(567) will not cause the Error Flag to go ON in these cases.

- The Negative Flag will turn ON if the MSB in the transferred data is ON.

Function

● Data Collection Operation

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.



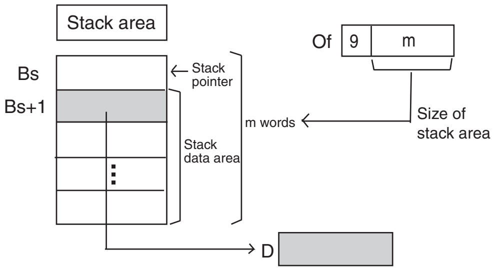
● Stack read operation

- FIFO Stack Read Operation

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.

FIFO
Data is copied from Bs + 1.

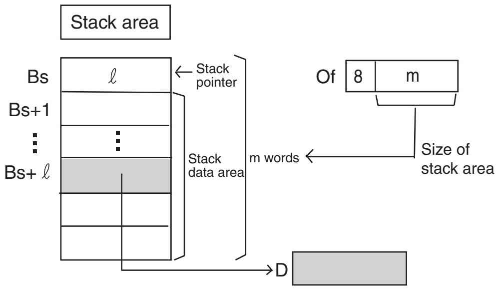


- LIFO Stack Read Operation

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.

LIFO
Data is copied from Bs + stack pointer.



In either case (method), S1 is a stack pointer.

- When FIFO is used, data is read from S1+1.
- When LILO is used, data is read from S1 + (stack pointer).
- In either case (method), the stack pointer value is automatically decremented by 1 each time data is read from the stack area.

Note 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.

Hint

- Use DISTC(566) to write stack data to the stack area.
- 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.

Precaution

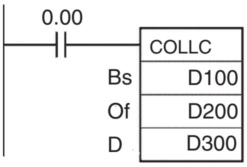
- Be sure that the offset or stack size specified by Of does not exceed the end of the data area when added to Bs.

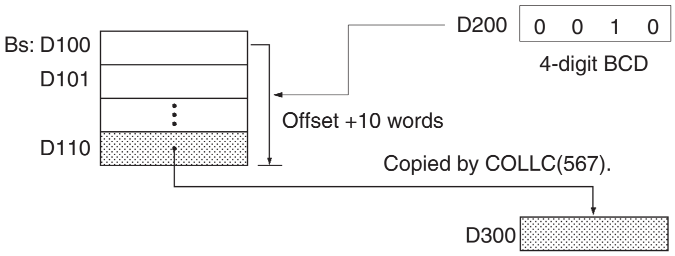
Example Programming

● 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.

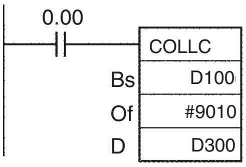


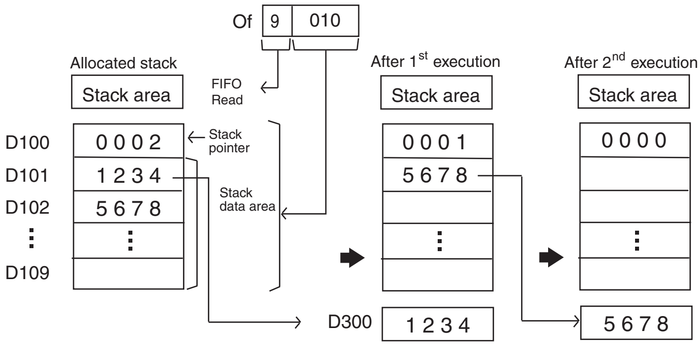


● FIFO Stack Operation

The leftmost byte of Of is 9, so COLLC(567) performs the FIFO Stack Operation.

When CIO 0.00 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 D101 (Bs +1) are copied to D300. Finally, the stack pointer is decremented by 1.

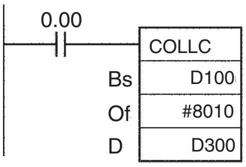


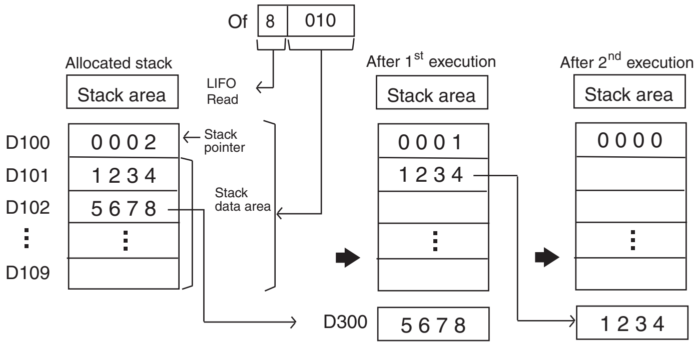


● LIFO Stack Operation

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.



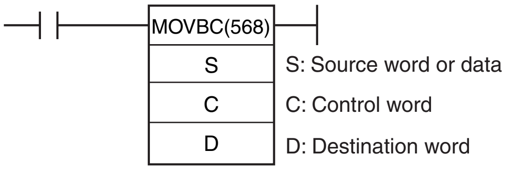


MOVBC

Instruction	Mnemonic	Variations	Function code	Function
MOVE BIT	MOVBC	@MOVBC	568	Transfers the specified bit.

Symbol

MOVBC



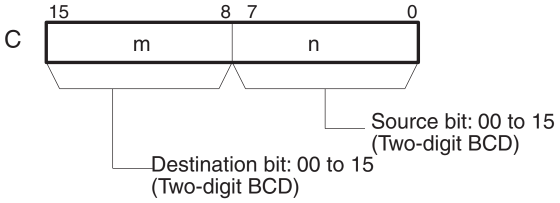
Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source word or data	WORD	1
C	Control word	WORD	1
D	Destination word	WORD	1

C: Control word



● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
C																		
D											---							

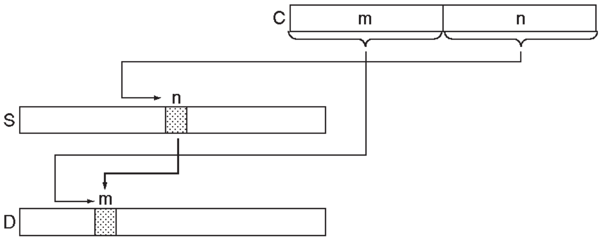
Flags

Name	Label	Operation
Error Flag	P_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.

Function

MOVBC(568) copies the specified bit (n) from S to the specified bit (m) in D.



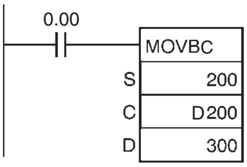
Note The other bits in the destination word are left unchanged.

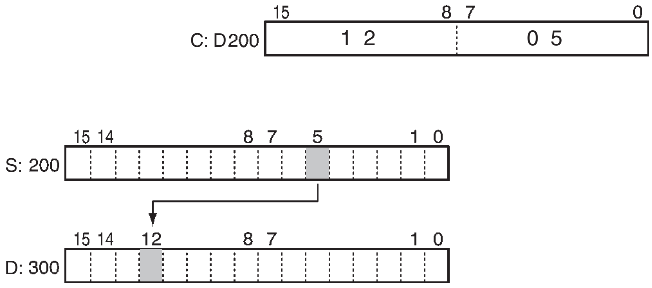
Hint

- The same word can be specified for both S and D to copy a bit within a word.

Example Programming

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.



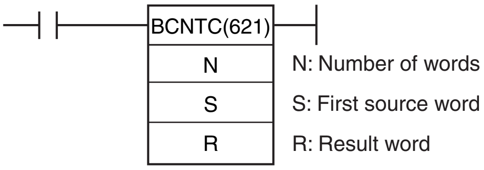


BCNTC

Instruction	Mnemonic	Variations	Function code	Function
BIT COUNTER	BCNTC	@BCNTC	621	Counts the total number of ON bits in the specified word(s).

Symbol

BCNTC



Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
N	Number of words	WORD	1
S	First source word	UNIT	Variable
R	Result word	WORD	1

N: Number of words

The number of words must be 0001 to 9999 (BCD).

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
N	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	---	---	---
S											---	---						
R												OK						

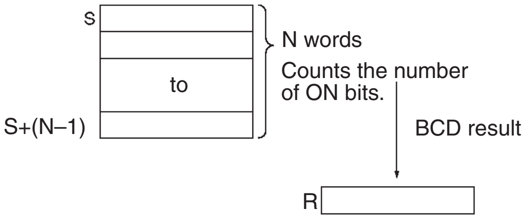
Flags

Name	Label	Operation
Error Flag	P_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	P_EQ	• 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.

Function

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.

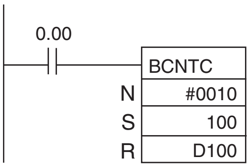


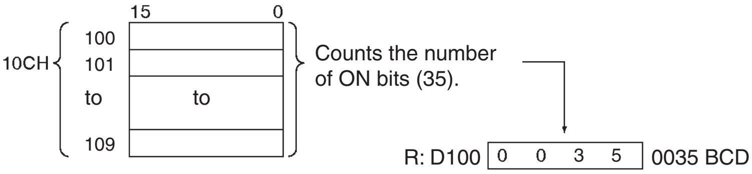
Precaution

- S and S+(N-1) must be in the same data area.

Example Programming

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 100 through CIO 109 and writes the result to D100.

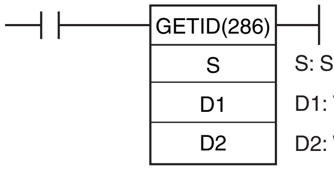




Special Function Block Instructions

GETID

Instruction	Mnemonic	Variations	Function code	Function
GET VARIABLE ID	GETID	@GETID	286	Outputs the FINS command variable type (data area) code and word address for the specified variable or address.

	GETID
Symbol	 S: Source data D1: Variable code D2: Word address

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
S	Source data	WORD	1
D1	Variable code	WORD	1
D2	Word address	WORD	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	---	OK	---	OK	---	---	---	---
D1																		
D2																		

Flags

Name	Label	Operation
Error Flag	P_ER	• ON if S is not within the allowed range. • OFF in all other cases.

Function

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.

- Variable type (data area)
- Corresponding address ranges

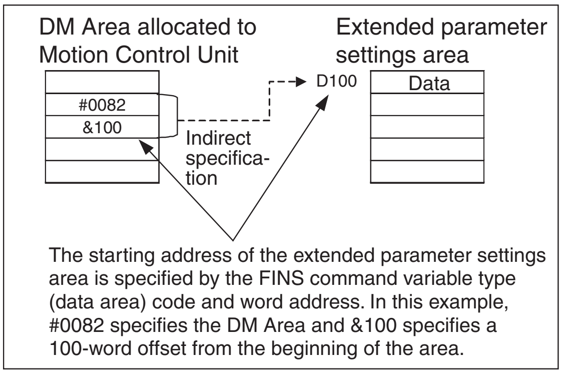
Data area		Data size	Data area code (Output to D1.)	Address (Output to D2.)
CIO Area	CIO	Word	00B0 hex	0000 to 17FF hex (0000 to 6143)
Work Area	W		00B1 hex	0000 to 01FF hex (000 to 511)
Holding Area	H		00B2 hex	0000 to 01FF hex (000 to 511)
DM Area	D		0082 hex	0000 to 7FFF hex (00000 to 32767)
EM Area (Specific bank)	En_ (n = 0 to 18 hex)		0050 to 0068 hex	0000 to 7FFF hex (00000 to 32767)

Hint

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.

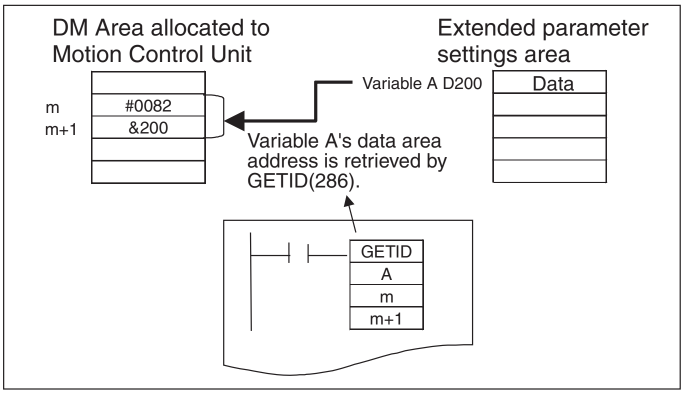
Example Programming

Normal Operation



The starting address of the extended parameter settings area is specified by the FINS command variable type (data area) code and word address. In this example, #0082 specifies the DM Area and &100 specifies a 100-word offset from the beginning of the area.

Using Function Blocks



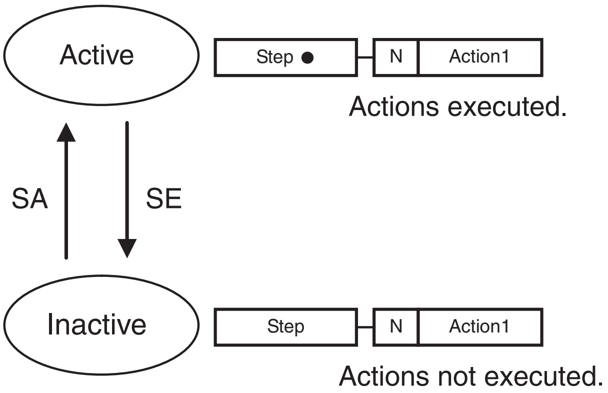
SFC Instructions

Step Control Instructions

Step Control Instructions are used to control the active status of steps and subcharts in SFC programs. There are two Step Control Instructions.

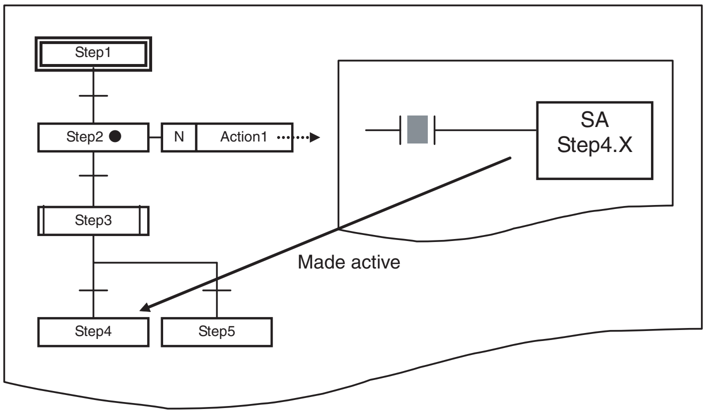
Instruction	Mnemonic	Outline
STEP ACTIVATION	SA	Makes the specified step or subchart active to start execution of the actions.
STEP DEACTIVATION	SE	Makes the specified step or subchart inactive to end execution of the actions.

Status is changed as follows by the Step Control Instructions.

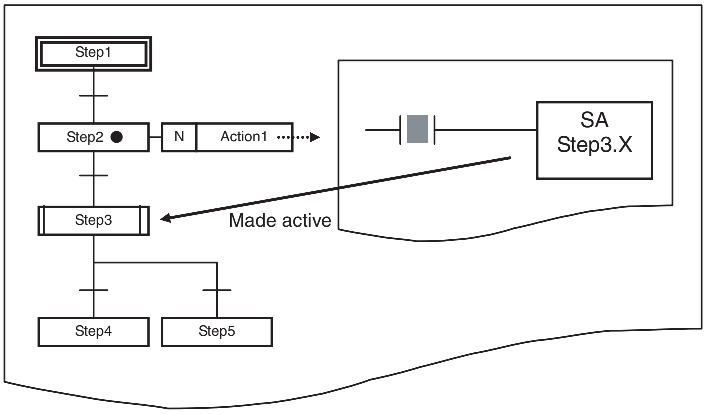


Note The active step is indicated by a solid circle (●).

Example 1: Action 1 in step 2 can be used to make step 4 active.



Example 2: A subchart step can be specified to make the subchart active.

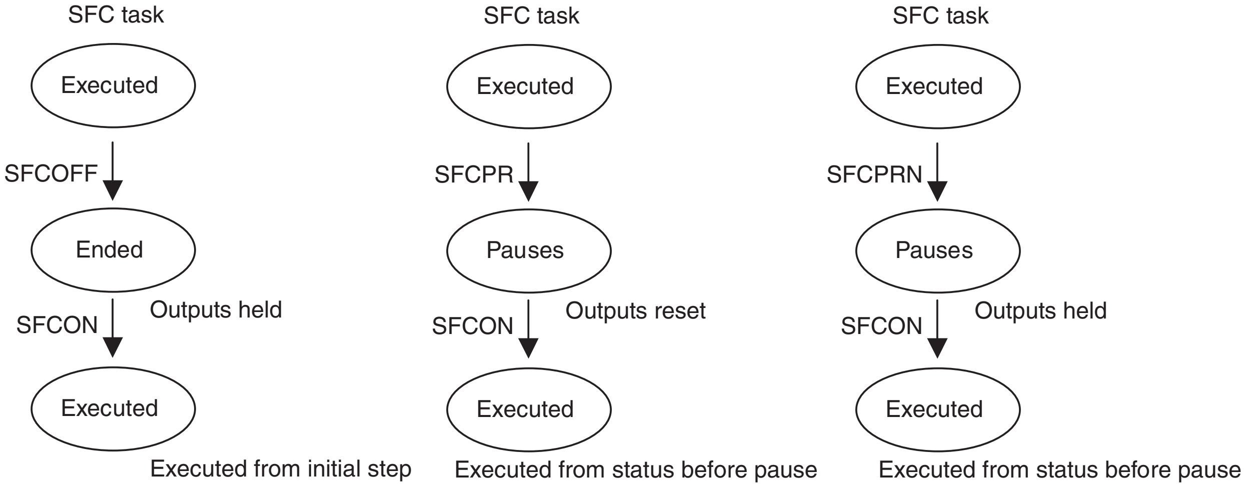


SFC Task Control Instruction

The following type of control can be achieved by using the SFC Task Control Instructions

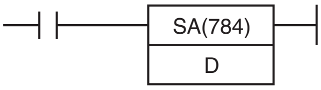
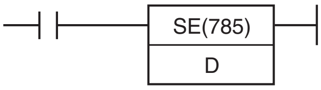
- Ending SFC task execution using the SFCOFF instruction
- Pausing SFC task execution and resetting outputs using the SFCPR instruction
- Pausing SFC task execution and holding outputs using the SFCPRN instruction
- Restarting execution of SFC tasks that have been ended or paused using the SFCON instruction

SFC OFF: SFCOFF	Ends execution of an SFC task. The outputs from any actions that are being executed in the task that is ended will be held and processing will be ended. When execution is started again with the SFCON instruction, it will start from the initial step.
SFC PAUSE WITH RESET: SFCPR	All outputs in the task that is paused will be reset and processing will be paused. Be careful when using this command if any outputs that will be reset are used in other tasks. When execution is started again with the SFCON instruction, it will start from the step that was active before the task was paused. The Step Timer for the paused step will continue to operate even while the step is paused.
SFC PAUSE WITH NO RESET: SFCPRN	Pauses execution of an SFC task. The outputs from any actions that are being executed in the task that is paused will be held and processing will be paused. When execution is started again with the SFCON instruction, it will start from the step that was active before the task was paused. The Step Timer for the paused step will continue to operate even while the step is paused.
SFC ON: SFCON	Restarts execution of an SFC task that was ended or paused using one of the other SFC Task Control Instructions.



SA/SE

Instruction	Mnemonic	Variations	Function code	Function
STEP ACTIVATE	SA	@SA	784	Makes the specified step or subchart active to start execution of the actions.
STEP DEACTIVATE	SE	@SE	785	Makes the specified step of subchart inactive to end execution of the actions.

Symbol	SA	SE
		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

Operand	Description	Data type	Size
D	Step flag	BOOL	1

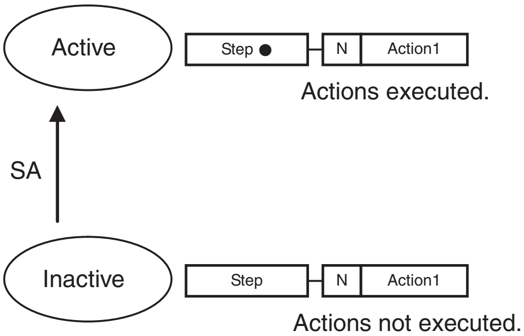
Flags

Name	Label	Operation
Error Flag	P_ER	OFF

Function

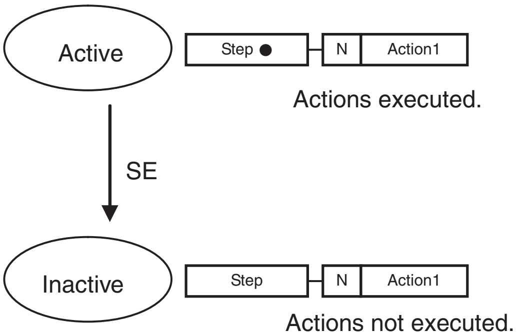
● SA(784)

The step specified by D is made active.



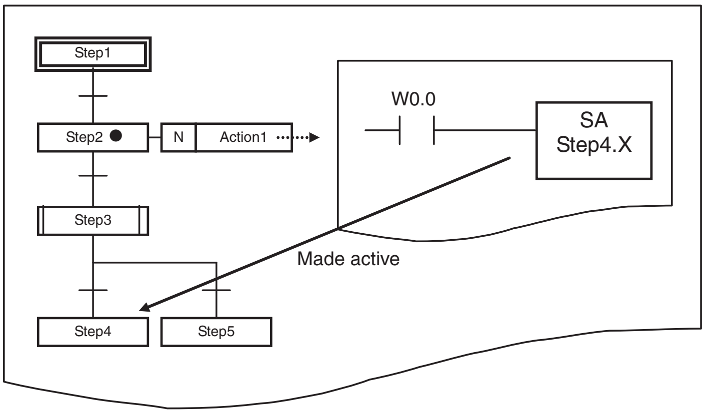
● SE(785)

The step designated by D is made inactive.

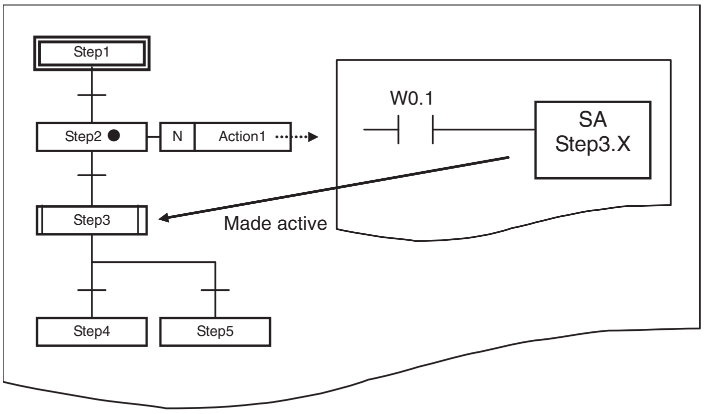


Example Programming

In the following example, step 4 will become active when W0.0 turns ON.



In the following example, step 3 (a subchart step) will become active when W0.1 turns ON.

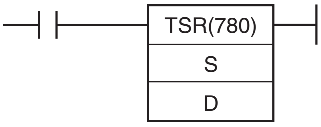
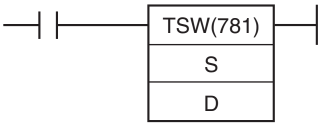


Precaution

- Designate Step Flags in the form step_name.X. Refer to the *SYSMAC CS/CJ Series Programmable Controllers Operation Manual SFC Programming* (W469) for details.
- Steps within subcharts cannot be specified.
- These instructions cannot be input or displayed on a Programming Console. “?” will be displayed.

TSR/TSW

Instruction	Mnemonic	Variations	Function code	Function
READ SET TIMER	TSR	@TSR	780	The present value of the Step Timer specified by S is stored starting at D.
SET STEP TIMER	TSW	@TSW	781	The present value of the Step Timer specified by S is changed to the value specified starting at D.

	TSR	TSW
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	OK	OK

Operands

● TSR

Operand	Description	Data type	Size
S	Step timer	UINT	1
D	First word to store the Step Timer's present value	UINT	1

● TSW

Operand	Description	Data type	Size
S	First word holding the Step Timer's new present value	UINT	1
D	Step timer	UINT	1

● Operand Specifications

Area	Word addresses								Indirect DM/EM addresses		Con-stants	Registers			Flags		Pulse bits	TR bits
	CIO	WR	HR	AR	T	C	DM	EM	@DM @EM	*DM *EM		DR	IR	Indirect using IR	TK	CF		
S	OK	OK	OK	OK	---	---	OK	OK	---	---	---	---	---	---	---	---	---	---
D											---	OK						

Flags

Name	Label	Operation
Error Flag	P_ER	OFF

Function

● TSR(780)

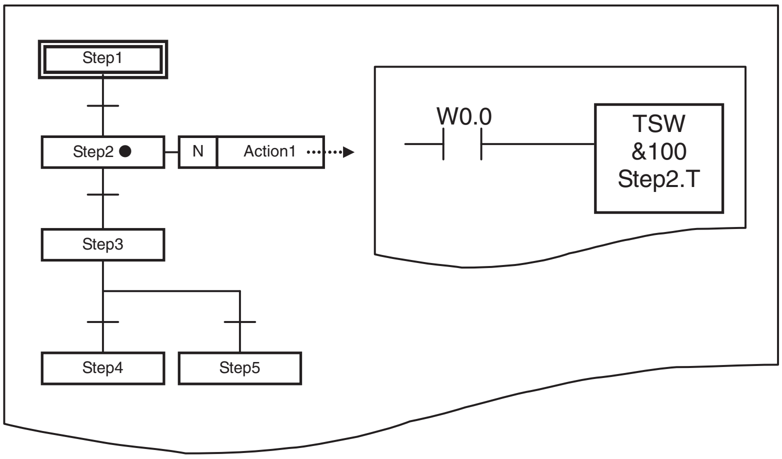
The present value of the Step Timer specified by S is stored starting at D.

● TSW(781)

The present value of the Step Timer specified by S is changed to the value specified starting at D.

Example Programming

In the following example, the present value of the Step Timer for step 2 is changed to 100 when W0.0 turns ON.



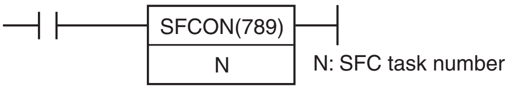
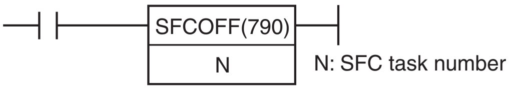
Note The active step is indicated by a solid circle (●).

Precaution

- Designate Step Flags in the form step_name.X. Refer to the *SYSMAC CS/CJ Series Programmable Controllers Operation Manual SFC Programming* (W469) for details.
- These instructions cannot be input or displayed on a Programming Console. “?” will be displayed.

SFCON/SFCOFF

Instruction	Mnemonic	Variations	Function code	Function
SFC ON	SFCON	---	789	Restarts execution of an SFC task that was ended or paused using one of the other SFC Task Control Instructions.
SFC OFF	SFCOFF	---	790	Ends execution of an SFC task. The status of all outputs is held. When execution of the SFC task is restarted, it is executed from the initial step.

	SFCON	SFCOFF
Symbol		

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
N	SFC task number	---	1

Cyclic Task

N: &0 to &31 (Correspond to cyclic tasks 0 to 31.) For CJ2 CPU Units, &0 to &127 are available.

Extra Task

N: &8000 to &8255 (Correspond to interrupt tasks 0 to 255.)

Operand	Task name
&0 to &127	Cyclic tasks 0 to 127
&8000	Extra task 0 (interrupt task 0)
&8001	Extra task 1 (interrupt task 1)
...	...
&8255	Extra task 255 (interrupt task 255)

Flags

Name	Label	Operation
Error Flag	P_ER	• OFF if the specified task does not exist. • OFF if the instruction is executed in the interrupt tasks.

Function

● SFCON(789)

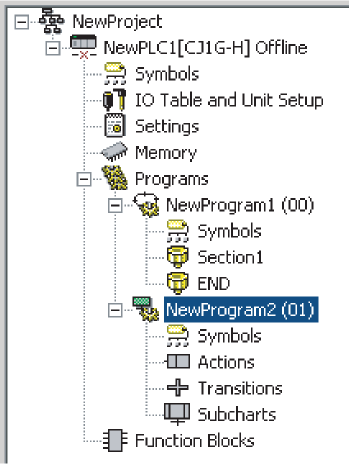
The SFC task specified by N is started. If an ended task is started, it is executed from the initial step. If a paused task is started, it is executed from the status that existed before it was paused.

● SFCOFF(790)

The SFC task specified by N is ended. The status of all outputs is held. When execution is restarted with the SFCON(789) instruction, the task is executed from the initial step.

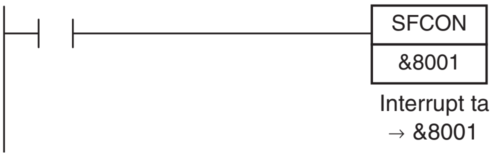
Precaution

- Specify the number displayed to the right of the SFC program in the workspace for the SFC task number, as shown below.



As shown below, SFC programs can be created in interrupt tasks and executed as extra tasks.

When using an extra task, specify the SFC task number as shown above for the operand data. For the program shown above for extra task 1 (interrupt task 1), the SFC task number specification would be as follows:

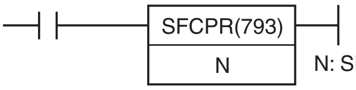
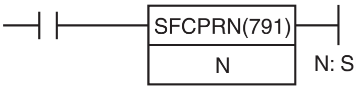


Interrupt task 01
→ &8001

- These instructions cannot be used for the task in which they are programmed. For example, if the instruction is in cyclic task 0 (task number 0), a task number of 0 cannot be specified.
- This instruction cannot be input or displayed on a Programming Console. “?” will be displayed.

SFCPR/SFCPRN

Instruction	Mnemonic	Variations	Function code	Function
SFC PAUSE WITH RESET	SFCPR	---	793	Pauses execution of an SFC task. The status of all outputs is reset. When execution of a paused task is restarted, execution will start from the step that was active when the task was paused.
SFC PAUSE WITH NO RESET	SFCPRN	---	791	Pauses execution of an SFC task. The status of all outputs is held. When execution of a paused task is restarted, execution will start from the step that was active when the task was paused.

	SFCPR	**SFCPRN**
Symbol	 N: SFC task number	 N: SFC task number

Applicable Program Areas

Area	Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks	SFC action or transition programs
Usage	OK	OK	OK	OK	Not allowed	OK

Operands

Operand	Description	Data type	Size
N	SFC task number	---	1

Cyclic Task

N: &0 to &31 (Correspond to cyclic tasks 0 to 31.) For CJ2 CPU Units, &0 to &127 are available.

Extra Task

N: &8000 to &8255 (Correspond to interrupt tasks 0 to 255.)

Operand	Task name
&0 to &127	Cyclic tasks 0 to 127
&8000	Extra task 0 (interrupt task 0)
&8001	Extra task 1 (interrupt task 1)
...	...
&8255	Extra task 255 (interrupt task 255)

Flags

Name	Label	Operation
Error Flag	P_ER	• OFF if the specified task does not exist. • OFF if the instruction is executed in the interrupt tasks

Function

● SFCPR(793)

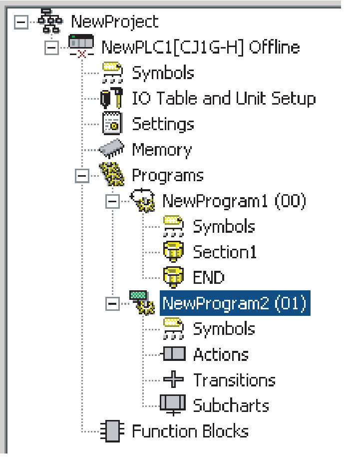
Execution of the SFC task specified by N is paused. All outputs from the paused task are reset. When the paused task is restarted with the SFCON(789) instruction, it is executed from the status that existed before it was paused.

● SFCPRN(791)

Execution of the SFC task specified by N is paused. All outputs from the paused task are held. When the paused task is restarted with the SFCON(789) instruction, it is executed from the status that existed before it was paused.

Precaution

- Specify the number displayed to the right of the SFC program in the workspace for the SFC task number, as shown below.



As shown below, SFC programs can be created in interrupt tasks and executed as extra tasks.

When using an extra task, specify the SFC task number as shown above for the operand data. For the program shown above for extra task 1 (interrupt task 1), the SFC task number specification would be as follows:

- These instructions cannot be used for the task in which they are programmed. For example, if the instruction is in cyclic task 0 (task number 0), a task number of 0 cannot be specified.
- This instruction cannot be input or displayed on a Programming Console. “?” will be displayed.
- The step timers will continue to operate for steps paused with the SFCPR or SFCPRN instruction. The displayed time, however, will not change.
- Step timers may not operate correctly when the SFCPR or SFCPRN instruction is used.

● SFCPR

- Do not execute the SFCPR instruction for an SFC task that has been ended (see note.). Even if the SFCPR instruction is executed for an SFC task that has been ended, the outputs will be reset and the task will not be started. If this happens, you can execute SFCON again to start the task.

Note Use the Task Flag to prevent the SFCPR instruction from being executed for an SFC task that has been ended. This is not possible, however, when using an SFC program in an extra task.

- If the SCFPR instruction is executed for a step for which status has changed in the same cycle, the outputs will be reset. This means that the final scan and the P0 action in steps that became inactive will not be executed.
- If the SCFPR instruction is executed for a step for which status has changed in the same cycle and then the SFCON instruction is executed, the SFC task will be started, but the P and P1 actions in steps that became active will not be executed. Refer to the SYSMAC CS/CJ Series Programmable Controllers Operation Manual SFC Programming (W469) for details.

SECTION 4
Instruction Execution Times and Number of Steps

This section provides instruction execution times and the number of steps for each CS/CJ-series instruction.

4-1 CJ2 CPU Unit Instruction Execution Times and Number of Steps

The following table lists the execution times for all instructions that are supported by the CPU Units.

The total execution time of instructions within one whole user program is the process time for program execution when calculating the cycle time (See note.).

Note User programs are allocated tasks that can be executed within cyclic tasks and interrupt tasks that satisfy interrupt conditions.

Execution times for most instructions differ depending on the CPU Unit used and the conditions when the instruction is executed.

The execution time can also vary when the execution condition is OFF.

The following table also lists the length of each instruction in the *Length (steps)* column. The number of steps required in the user program area for each instructions depends on 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. Most instructions are supported in differentiated form (indicated with ↑, ↓, @, and %). Specifying differentiation will increase the execution times by the following amounts. (unit: µs)

Symbol	CJ2H	CJ2M
	CPU6□-EIP	CPU3□
↑ or ↓	+0.24	+0.32
@ or %	+0.24	+0.32

2. Use the following time as a guideline when instructions are not executed. (unit: µs)

CJ2H	CJ2M
CPU6□-EIP	CPU□□
0.016 to 0.096	0.020 to 0.120

4-1-1 Sequence Input Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
LOAD	LD	1 to 2	0.016	0.040	---
	!LD	2 to 14	0.99	1.260	---
LOAD NOT	LD NOT	1 to 2	0.016	0.040	---
	!LD NOT	2 to 14	0.99	1.260	---
AND	AND	1 to 2	0.016	0.040	---
	!AND	2 to 14	0.99	1.260	---
AND NOT	AND NOT	1 to 2	0.016	0.040	---
	!AND NOT	2 to 14	0.99	1.260	---
OR	OR	1 to 2	0.016	0.040	---
	!OR	2 to 14	0.99	1.260	---
OR NOT	OR NOT	1 to 2	0.016	0.040	---
	!OR NOT	2 to 14	0.99	1.260	---
AND LOAD	AND LD	1	0.016	0.040	---
OR LOAD	OR LD	1	0.016	0.040	---
NOT	NOT	1	0.016	0.040	---
CONDITION ON	UP	3	0.26	0.36	---
CONDITION OFF	DOWN	4	0.27	0.40	---
LOAD BIT TEST	LD TST	4	0.11	0.16	---
LOAD BIT TEST NOT	LD TSTN	4	0.11	0.16	---
AND BIT TEST	AND TST	4	0.11	0.16	---
AND BIT TEST NOT	AND TSTN	4	0.11	0.16	---
OR BIT TEST	OR TST	4	0.11	0.16	---
OR BIT TEST NOT	OR TSTN	4	0.11	0.16	---

4-1-2 Sequence Output Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
OUTPUT	OUT	1 to 2	0.016	0.040	---
	!OUT	2 to 14	0.99	1.320	---
OUTPUT NOT	OUT NOT	1 to 2	0.016	0.040	---
	!OUT NOT	2 to 14	0.99	1.320	---
KEEP	KEEP	1 to 2	0.048	0.060	---
	!KEEP	16	0.99	1.340	---
DIFFERENTIATE UP	DIFU	2 to 2	0.28	0.30	---
DIFFERENTIATE DOWN	DIFD	2 to 2	0.24	0.30	---
SET	SET	1 to 2	0.016	0.040	---
	!SET	2 to 14	0.99	1.360	---
RESET	RSET	1 to 2	0.016	0.040	---
	!RSET	2 to 14	0.99	1.360	---
MULTIPLE BIT SET	SETA	4	3.68	4.12	With 1-bit set
			15.5	24.4	With 1,000-bit set
MULTIPLE BIT RESET	RSTA	4	3.7	4.1	With 1-bit reset
			15.5	24.4	With 1,000-bit reset
SINGLE BIT SET	SETB	2	0.19	0.280	---
	!SETB	16	0.99	1.120	---
SINGLE BIT RESET	RSTB	2	0.19	0.280	---
	!RSTB	16	0.99	1.120	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SINGLE BIT OUTPUT	OUTB	2	0.19	0.280	---
	!OUTB	16	0.99	1.180	---

4-1-3 Sequence Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
END	END	1	2.6	3.5	---
NO OPERATION	NOP	1	0.016	0.040	---
INTERLOCK	IL	1	0.048	0.060	---
INTERLOCK CLEAR	ILC	1	0.048	0.060	---
MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	3	2.3	3.3	Interlock condition not met (input condition ON)
			3.4	4.6	Interlock condition met (input condition OFF)
			3.8	5.2	Interlock condition met again during interlock (input condition OFF)
MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	3	2.3	3.1	Interlock condition not met (input condition ON)
			3.4	4.5	Interlock condition met (input condition OFF)
			3.8	5.1	Interlock condition met again during interlock (input condition OFF)
MULTI-INTERLOCK CLEAR	MILC	2	1.2	1.7	Not during interlock
			1.6	2.2	During interlock
JUMP	JMP	2	0.31	0.34	---
JUMP END	JME	2	---	---	---
CONDITIONAL JUMP	CJP	2	0.31	0.34	Jump condition met (input condition ON)
CONDITIONAL JUMP NOT	CJPN	2	0.31	0.34	Jump condition met (input condition OFF)
MULTIPLE JUMP	JMP0	1	0.048	0.060	---
MULTIPLE JUMP END	JME0	1	0.048	0.060	---
FOR LOOP	FOR	2	0.27	0.42	Designating a constant
BREAK LOOP	BREAK	1	0.048	0.060	---
NEXT LOOP	NEXT	1	0.14	0.16	When loop is continued
			0.18	0.18	When loop is ended

4-1-4 Timer and Counter Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
HUNDRED-MS TIMER	TIM	3	0.67	0.84	---
	TIMX		0.67	0.84	

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
TEN-MS TIMER	TIMH	3	0.67	0.84	---
	TIMHX		0.67	0.84	
ONE-MS TIMER	TMHH	3	0.67	0.84	---
	TMHHX		0.67	0.84	
TENTH-MS TIMER	TIMU	3	0.67	0.84	---
	TIMUX		0.67	0.84	
HUNDREDTH-MS TIMER	TMUH	3	0.67	0.84	---
	TMUHX		0.67	0.84	
ACCUMULATIVE TIMER	TTIM	3	9.2	12.1	---
			6.9	8.4	When resetting
			5.0	6.5	When interlocking
	TTIMX	3	8.8	11.7	---
			6.8	8.5	When resetting
			5.0	6.5	When interlocking
LONG TIMER	TIML	4 to 5	5.8	7.0	---
			3.9	4.1	When interlocking
	TIMLX	4 to 5	5.7	7.0	---
			3.6	3.7	When interlocking
MULTI-OUTPUT TIMER	MTIM	4	6.4	7.2	---
			3.7	4.3	When resetting
	MTIMX	4	5.5	6.4	---
			3.4	3.8	When resetting
TIMER RESET	TRSET	2	0.58	0.8	---
COUNTER	CNT	3	0.51	0.58	---
	CNTX		0.51	0.58	
REVERSIBLE COUNTER	CNTR	3	9.1	11.8	---
	CNTRX		8.0	10.3	
RESET TIMER/COUNTER	CNR	3	4.8	5.4	When resetting 1 word
			2839	2555	When resetting 1,000 words
	CNRX	3	4.7	5.5	When resetting 1 word
			2839	2555	When resetting 1,000 words

4-1-5 Comparison Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
Input Comparison Instructions (unsigned)	=	4	0.08	0.16	---
	<>				
	<				
	<=				
	>				
	>=				

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
Input Comparison Instructions (double, unsigned)	=L	4 to 7	0.08	0.24	---
	<>L				---
	<L				---
	<=L				---
	>L				---
	>=L				---
Input Comparison Instructions (signed)	=S	4	0.08	0.16	---
	<>S				
	<S				
	<=S				
	>S				
	>=S				
Input Comparison Instructions (double, signed)	=SL	4 to 7	0.08	0.24	---
	<>SL				
	<SL				
	<=SL				
	>SL				
	>=SL				
Time Comparison Instructions	=DT	4	16.300	27.9	---
	<>DT				
	<DT				
	<=DT				
	>DT				
	>=DT				
COMPARE	CMP	3	0.06	0.080	---
	!CMP	30	2.06	2.6	---
DOUBLE COMPARE	CMPL	3 to 5	0.064	0.120	---
SIGNED BINARY COMPARE	CPS	3 to 5	0.064	0.080	---
	!CPS	30	2.06	2.6	---
DOUBLE SIGNED BINARY COMPARE	CPSL	3 to 5	0.064	0.120	---
TABLE COMPARE	TCMP	4	10.3	12.5	---
MULTIPLE COMPARE	MCMP	4	15.2	20.3	---
UNSIGNED BLOCK COMPARE	BCMP	4	16.3	20.5	---
EXPANDED BLOCK COMPARE	BCMP2	4	5.0	5.1	Number of data words: 1
			217.2	278	Number of data words: 255
AREA RANGE COMPARE	ZCP	3	0.14	0.400	---
DOUBLE AREA RANGE COMPARE	ZCPL	3 to 5	0.14	0.640	---
SIGNED AREA RANGE COMPARE	ZCPS	3	0.14	0.400	
DOUBLE SIGNED AREA RANGE COMPARE	ZCPSL	3 to 5	0.14	0.640	

4-1-6 Data Movement Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
MOVE	MOV	3	0.05	0.12	---
	!MOV	30	1.98	2.6	---
DOUBLE MOVE	MOVL	3 to 4	0.05	0.20	---
MOVE NOT	MVN	3	0.05	0.12	---
DOUBLE MOVE NOT	MVNL	3 to 4	0.05	0.20	---
MOVE BIT	MOVB	4	0.19	0.32	---
MOVE DIGIT	MOVD	4	0.19	0.32	---
MULTIPLE BIT TRANSFER	XFRB	4	6.6	9.4	Transferring 1 bit
			85.8	119	Transferring 255 bits
BLOCK TRANSFER	XFER	4	0.29	0.28	Transferring 1 word
			240.1	220	Transferring 1,000 words
BLOCK SET	BSET	4	0.21	0.20	Setting 1 word
			142.2	140	Setting 1,000 words
DATA EXCHANGE	XCHG	3	0.32	0.48	---
DOUBLE DATA EXCHANGE	XCGL	3 to 4	0.12	0.29	---
SINGLE WORD DISTRIBUTE	DIST	4	4.5	4.7	---
DATA COLLECT	COLL	4	4.6	4.7	---
MOVE TO REGISTER	MOVR	3	0.064	0.200	---
MOVE TIMER/COUNTER PV TO REGISTER	MOVRW	3	0.064	0.200	---

4-1-7 Data Shift Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SHIFT REGISTER	SFT	3	2.86	3.47	Shifting 1 word
			315	422	Shifting 1,000 words
REVERSIBLE SHIFT REGISTER	SFTR	4	6.22	6.38	Shifting 1 word
			319	422	Shifting 1,000 words
ASYNCHRONOUS SHIFT REGISTER	ASFT	4	5.3	6.3	Shifting 1 word
			948	1285	Shifting 1,000 words[*1]
WORD SHIFT	WSFT	4	2.3	3.1	Shifting 1 word
			233	187	Shifting 1,000 words
ARITHMETIC SHIFT LEFT	ASL	2	0.18	0.260	---
DOUBLE SHIFT LEFT	ASLL	2	0.32	0.420	---
ARITHMETIC SHIFT RIGHT	ASR	2	0.18	0.260	---
DOUBLE SHIFT RIGHT	ASRL	2	0.32	0.420	---
ROTATE LEFT	ROL	2	0.18	0.260	---
DOUBLE ROTATE LEFT	ROLL	2	0.32	0.420	---
ROTATE LEFT WITHOUT CARRY	RLNC	2	0.18	0.260	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	2	0.32	0.420	---
ROTATE RIGHT	ROR	2	0.18	0.260	---
DOUBLE ROTATE RIGHT	RORL	2	0.32	0.420	---
ROTATE RIGHT WITHOUT CARRY	RRNC	2	0.18	0.260	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	2	0.32	0.420	---
ONE DIGIT SHIFT LEFT	SLD	3	3.7	4.4	Shifting 1 word
			317.8	429	Shifting 1,000 words
ONE DIGIT SHIFT RIGHT	SRD	3	4.5	5.4	Shifting 1 word
			479.5	656	Shifting 1,000 words
SHIFT N-BIT DATA LEFT	NSFL	4	4.6	5.2	Shifting 1 bit
			31.5	36.1	Shifting 1,000 bits
SHIFT N-BIT DATA RIGHT	NSFR	4	4.5	5.2	Shifting 1 bit
			39.0	50.2	Shifting 1,000 bits
SHIFT N-BITS LEFT	NASL	3	0.18	0.38	---
DOUBLE SHIFT N-BITS LEFT	NSLL	3	0.32	0.54	---
SHIFT N-BITS RIGHT	NASR	3	0.18	0.38	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	3	0.32	0.54	---

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-1-8 Increment/Decrement Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
INCREMENT BINARY	++	2	0.18	0.24	---
DOUBLE INCREMENT BINARY	++L	2	0.18	0.24	---
DECREMENT BINARY	– –	2	0.18	0.24	---
DOUBLE DECREMENT BINARY	– –L	2	0.18	0.24	---
INCREMENT BCD	++B	2	3.0	3.4	---
DOUBLE INCREMENT BCD	++BL	2	3.2	3.5	---
DECREMENT BCD	– –B	2	3.0	3.5	---
DOUBLE DECREMENT BCD	– –BL	2	3.2	3.5	---

4-1-9 Symbol Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SIGNED BINARY ADD WITHOUT CARRY	+	4	0.18	0.34	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	4 to 6	0.18	0.24	---
SIGNED BINARY ADD WITH CARRY	+C	4	0.18	0.34	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	4 to 6	0.18	0.24	---
BCD ADD WITHOUT CARRY	+B	4	4.0	4.8	---
DOUBLE BCD ADD WITHOUT CARRY	+BL	4 to 6	4.9	6.0	---
BCD ADD WITH CARRY	+BC	4	4.4	5.2	---

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
DOUBLE BCD ADD WITH CARRY	+BCL	4 to 6	5.2	6.6	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	4	0.18	0.340	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	4 to 6	0.18	0.24	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	4	0.18	0.340	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	4 to 6	0.18	0.24	---
BCD SUBTRACT WITHOUT CARRY	–B	4	4.1	4.9	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	4 to 6	4.9	5.9	---
BCD SUBTRACT WITH CARRY	–BC	4	4.5	5.2	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	4 to 6	5.2	6.3	---
SIGNED BINARY MULTIPLY	*	4	0.26	0.520	---
DOUBLE SIGNED BINARY MULTIPLY	*L	4 to 6	3.6	3.9	---
UNSIGNED BINARY MULTIPLY	*U	4	0.26	0.26	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	4 to 6	3.6	3.9	---
BCD MULTIPLY	*B	4	3.6	4.6	---
DOUBLE BCD MULTIPLY	*BL	4 to 6	4.9	6.2	---
SIGNED BINARY DIVIDE	/	4	0.29	0.540	---
DOUBLE SIGNED BINARY DIVIDE	/L	4 to 6	4.2	4.8	---
UNSIGNED BINARY DIVIDE	/U	4	0.29	0.540	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	4 to 6	3.8	4.2	---
BCD DIVIDE	/B	4	5.0	5.9	---
DOUBLE BCD DIVIDE	/BL	4 to 6	4.8	5.9	---

4-1-10 Conversion Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
BCD TO BINARY	BIN	3	0.18	0.280	---
DOUBLE BCD TO DOUBLE BINARY	BINL	3 to 4	3.3	3.5	---
BINARY TO BCD	BCD	3	0.19	0.300	---
DOUBLE BINARY TO DOUBLE BCD	BCDL	3 to 4	3.3	3.7	---
2’S COMPLEMENT	NEG	3	0.14	0.240	---
DOUBLE 2’S COMPLEMENT	NEGL	3 to 4	0.26	0.440	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN	3	0.26	0.340	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
DATA DECODER	MLPX	4	0.17	0.280	Decoding 1 digit (4 to 16)
			0.42	0.770	Decoding 4 digits (4 to 16)
			1.14	1.760	Decoding 1 digit (8 to 256)
			2.17	3.370	Decoding 4 digits (8 to 256)
DATA ENCODER	DMPX	4	3.3	4.6	Encoding 1 digit (16 to 4)
			3.7	5.2	Encoding 4 digits (16 to 4)
			17.3	26.3	Encoding 1 digit (256 to 8)
			35	47	Encoding 2 digits (256 to 8)
ASCII CONVERT	ASC	4	4.0	4.5	Converting 1 digit into ASCII
			4.6	5.2	Converting 4 digits into ASCII
ASCII TO HEX	HEX	4	3.3	3.8	Converting 1 digit
COLUMN TO LINE	LINE	4	10.5	13.1	---
LINE TO COLUMN	COLM	4	13.8	17.6	---
SIGNED BCD TO BINARY	BINS	4	3.6	4.0	Data format setting No. 0
			3.6	4.0	Data format setting No. 1
			3.6	4.0	Data format setting No. 2
			3.6	4.0	Data format setting No. 3
DOUBLE SIGNED BCD TO BINARY	BISL	4 to 5	3.7	4.1	Data format setting No. 0
			3.6	4.1	Data format setting No. 1
			3.7	4.2	Data format setting No. 2
			3.7	4.2	Data format setting No. 3
SIGNED BINARY TO BCD	BCDS	4	3.7	4.0	Data format setting No. 0
			3.7	4.1	Data format setting No. 1
			3.7	4.2	Data format setting No. 2
			3.7	4.2	Data format setting No. 3
DOUBLE SIGNED BINARY TO BCD	BDSL	4 to 5	4.0	4.5	Data format setting No. 0
			4.0	4.6	Data format setting No. 1
			4.0	4.6	Data format setting No. 2
			4.1	4.6	Data format setting No. 3

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
GRAY CODE CONVERSION	GRY	4	26.5	49.1	8-bit binary
			27.6	51.1	8-bit BCD
			30.9	57.2	8-bit angle
			35.3	66.0	15-bit binary
			36.3	68.0	15-bit BCD
			39.6	74.0	15-bit angle
GRAY CODE TO BINARY CONVERT	GRAY_BIN	3	0.1	0.3	---
DOUBLE GRAY CODE TO BINARY CONVERT	GRAY_BINL	3 to 4	0.1	0.4	---
DOUBLE GRAY CODE TO BINARY CONVERT	BIN_GRAY	3	0.1	0.3	---
DOUBLE BINARY TO GRAY CODE CONVERT	BIN_GRAYL	3 to 4	0.1	0.4	---
FOUR-DIGIT NUMBER TO ASCII	STR4	3	8.4	14.2	---
EIGHT-DIGIT NUMBER TO ASCII	STR8	3 to 4	10.2	16.4	---
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	3	15.8	28.2	---
ASCII TO FOUR-DIGIT NUMBER	NUM4	3 to 4	10.5	18.5	---
ASCII TO EIGHT-DIGIT NUMBER	NUM8	3	14.8	27.1	---
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	3	27.4	52.0	---

4-1-11 Logic Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
LOGICAL AND	ANDW	4	0.14	0.340	---
DOUBLE LOGICAL AND	ANDL	4 to 6	0.26	0.640	---
LOGICAL OR	ORW	4	0.18	0.340	---
DOUBLE LOGICAL OR	ORWL	4 to 6	0.26	0.640	---
EXCLUSIVE OR	XORW	4	0.18	0.340	---
DOUBLE EXCLUSIVE OR	XORL	4 to 6	0.26	0.640	---
EXCLUSIVE NOR	XNRW	4	0.18	0.340	---
DOUBLE EXCLUSIVE NOR	XNRL	4 to 6	0.26	0.640	---
COMPLEMENT	COM	2	0.18	0.240	---
DOUBLE COMPLEMENT	COML	2	0.32	0.440	---

4-1-12 Special Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
BINARY ROOT	ROTB	3	15.4	24.2	---
BCD SQUARE ROOT	ROOT	3	17.1	25.3	---
ARITHMETIC PROCESS	APR	4	4.6	5.3	Designating SIN and COS
			5.7	6.9	Designating line-segment approximation (with two coordinates in the line-segment data table)
FLOATING POINT DIVIDE	FDIV	4	76	149	---
BIT COUNTER	BCNT	4	0.24	0.360	Counting 1 word

4-1-13 Floating-point Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
FLOATING TO 16-BIT	FIX	3 to 4	0.13	0.24	---
FLOATING TO 32-BIT	FIXL	3 to 4	0.13	0.32	---
16-BIT TO FLOATING	FLT	3 to 4	0.13	0.30	---
32-BIT TO FLOATING	FLTL	3 to 4	0.13	0.32	---
FLOATING-POINT ADD	+F	4 to 6	0.24	0.66	---
FLOATING-POINT SUBTRACT	–F	4 to 6	0.24	0.66	---
FLOATING-POINT DIVIDE	/F	4 to 6	0.4	0.9	---
FLOATING-POINT MULTIPLY	*F	4 to 6	0.24	0.66	---
DEGREES TO RADIANS	RAD	3 to 4	2.7	3.3	---
RADIANS TO DEGREES	DEG	3 to 4	3.0	3.2	---
SINE	SIN	3 to 4	3.8	4.3	0° specified
			4.5	5.4	45° specified
			5.0	6.0	90° specified
HIGH-SPEED SINE	SINQ	8 to 9	0.59	0.86	0°, 45°, or 90° specified
COSINE	COS	3 to 4	3.7	4.3	0° specified
			4.4	5.2	45° specified
			5.3	6.7	90° specified
HIGH-SPEED COSINE	COSQ	8 to 9	0.59	0.86	0°, 45°, or 90° specified
TANGENT	TAN	3 to 4	3.9	4.5	0° specified
			6.1	8.2	45° specified
HIGH-SPEED TANGENT	TANQ	15 to 16	1.2	1.7	0°, 45°, or 90° specified
ARC SINE	ASIN	3 to 4	5.8	7.1	0° specified
			24.8	33.0	45° specified
			5.6	7.0	90° specified
ARC COSINE	ACOS	3 to 4	5.3	6.8	0° specified
			27.2	34.6	45° specified
			6.4	7.1	90° specified
ARC TANGENT	ATAN	3 to 4	4.0	5.0	0° specified
			5.6	7.0	45° specified
SQUARE ROOT	SQRT	3 to 4	0.42	0.66	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
EXPONENT	EXP	3 to 4	3.8	4.5	---
LOGARITHM	LOG	3 to 4	5.8	6.5	---
EXPONENTIAL POWER	PWR	4 to 6	35.7	56.6	---
Floating Symbol Comparison	=F	4 to 6	0.13	0.26	---
	<>F				
	<F				
	<=F				
	>F				
	>=F				
FLOATING- POINT TO ASCII	FSTR	4 to 5	15.6	23.9	---
ASCII TO FLOATING-POINT	FVAL	3	21.2	31.4	---
MOVE FLOATING-POINT (SINGLE)	MOVF	3 to 4	0.18	0.20	---

4-1-14 Double-precision Floating-point Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
DOUBLE SYMBOL COMPARISON	=D	4	5.1	6.7	---
	<>D				
	<D				
	<=D				
	>D				
	>=D				
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	3	5.1	5.4	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	3	5.1	5.4	---
16-BIT BINARY TO DOUBLE FLOATING	DBL	3	3.5	4.3	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	3	3.5	4.3	---
DOUBLE FLOATING-POINT ADD	+D	4	6.0	7.1	---
DOUBLE FLOATING-POINT SUBTRACT	–D	4	6.1	7.1	---
DOUBLE FLOATING-POINT MULTIPLY	*D	4	6.1	7.1	---
DOUBLE FLOATING-POINT DIVIDE	/D	4	6.4	7.5	---
DOUBLE DEGREES TO RADIANS	RADD	3	6.1	6.5	---
DOUBLE RADIANS TO DEGREES	DEGD	3	6.0	6.4	---
DOUBLE SINE	SIND	3	14.7	21.5	0° specified
			20.4	35.4	45° specified
			18.5	35.0	90° specified
DOUBLE COSINE	COSD	3	14.1	20.6	0° specified
			19.6	29.9	45° specified
			19.1	29.8	90° specified
DOUBLE TANGENT	TAND	3	7.3	9.4	0° specified
			27.4	50.3	45° specified

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
DOUBLE ARC SINE	ASIND	3	7.5	9.8	0° specified
			55.0	75.2	45° specified
			6.1	8.3	90° specified
DOUBLE ARC COSINE	ACOSD	3	8.3	10.9	0° specified
			55.9	72.8	45° specified
			43.7	72.8	90° specified
DOUBLE ARC TANGENT	ATAND	3	6.1	7.4	0° specified
			29.7	36.5	45° specified
DOUBLE SQUARE ROOT	SQRTD	3	16.6	23.4	---
DOUBLE EXPONENT	EXPD	3	39.7	58.4	---
DOUBLE LOGARITHM	LOGD	3	35.5	52.2	---
DOUBLE EXPONENTIAL POWER	PWRD	4	66	99	---

4-1-15 Table Data Processing Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SET STACK	SSET	3	7.6	9.4	Designating 5 words in stack area
			107	65	Designating 1,000 words in stack area
PUSH ONTO STACK	PUSH	3	4.9	5.9	---
FIRST IN FIRST OUT	FIFO	3	4.8	5.0	Designating 5 words in stack area
			231	167	Designating 1,000 words in stack area
LAST IN FIRST OUT	LIFO	3	5.3	7.1	---
DIMENSION RECORD TABLE	DIM	5	11.1	19.7	---
SET RECORD LOCATION	SETR	4	3.8	5.5	---
GET RECORD NUMBER	GETR	4	4.6	7.9	---
DATA SEARCH	SRCH	4	13.9	25.0	Searching for 1 word
			1940	3257	Searching for 1,000 words[*1]
SWAP BYTES	SWAP	3	10.1	17.5	Swapping 1 word
			1421	2098	Swapping 1,000 words[*1]
FIND MAXIMUM	MAX	4 to 5	4.8	5.8	Number of values being searched: 1
			465	672	Number of values being searched: 1,000[*1]
DOUBLE FIND MAXIMUM	MAXL	4 to 5	4.8	5.8	Number of values being searched: 1
			465	773	Number of values being searched: 1,000[*1]

Instruction	**Mnemonic**	**Length (steps)**	**ON execution time (μs)**		**Conditions**
			CJ2H CPU6□(-EIP)	**CJ2M CPU□□**	
FIND MAXIMUM FLOATING	MAXF	4 to 5	5.2	6.5	Number of values being searched: 1
			682	1090	Number of values being searched: 1,000*1
FIND DOUBLE MAXIMUM FLOATING	MAXD	4 to 5	5.4	6.4	Number of values being searched: 1
			1435	2333	Number of values being searched: 1,000*1
FIND MINIMUM	MIN	4 to 5	4.8	5.8	Number of values being searched: 1
			465	677	Number of values being searched: 1,000*1
DOUBLE FIND MINIMUM	MINL	4 to 5	4.8	5.9	Number of values being searched: 1
			189	774	Number of values being searched: 1,000*1
FIND MINIMUM FLOATING	MINF	4 to 5	5.2	6.5	Number of values being searched: 1
			683	1091	Number of values being searched: 1,000*1
FIND DOUBLE MINIMUM FLOATING	MIND	4 to 5	5.2	6.4	Number of values being searched: 1
			1402	2303	Number of values being searched: 1,000*1
SUM	SUM	4 to 5	17.5	31.3	Adding 1 word
			900	1696	Adding 1,000 words*1
FRAME CHECKSUM	FCS	4 to 5	14.1	25.2	For 1-word table length
			1235	2089	For 1,000-word table length*1
STACK SIZE READ	SNUM	3	4.5	5.3	---
STACK DATA READ	SREAD	4	4.6	5.4	---
STACK DATA OVERWRITE	SWRIT	4	4.3	5.0	---
STACK DATA INSERT	SINS	4	8.2	9.3	---
			275	256	For 1,000-word table
STACK DATA DELETE	SDEL	4	6.1	7.8	---
			247	180	For 1,000-word table

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-1-16 Tracking Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
Unsigned One-word Record Search Instructions	RSRCH	6	13.9	15.9	Number of records: 1
			504	585	Number of records: 1,000
Unsigned Two-word Record Search Instructions	RSRCH2	6	14.7	17.6	Number of records: 1
			838	932	Number of records: 1,000
Unsigned Four-word Record Search Instructions	RSRCH4	6	17.0	19.1	Number of records: 1
			1544	1684	Number of records: 1,000
UNSIGNED ONE-WORD RECORD SORT	RSORT	5	149	156	100 records, split sorting disabled, sorting “99, 98, 97...0" to “0, 1, 2...99" (worst-case scenario)
UNSIGNED TWO-WORD RECORD SORT	RSORT2	5	250	249	
UNSIGNED FOUR-WORD RECORD SORT	RSORT4	5	457	440	

4-1-17 Data Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
PID CONTROL	PID	4	297	526	First execution
			234	423	Input ON and sampling
			71	117	Input ON and not sampling
			7.2	10.5	First execution
PID CONTROL WITH AUTO-TUNING	PIDAT	4	302	600	Input ON and sampling
			237	428	Input ON and not sampling
			73	118	Initial execution of autotuning
			7.3	10.5	Autotuning when sampling
			120	203	Initial execution of autotuning
LIMIT CONTROL	LMT	4 to 5	10.8	18.3	---
DEAD BAND CONTROL	BAND	4 to 5	11.2	19.2	---
DEAD ZONE CONTROL	ZONE	4 to 5	10.9	17.7	---
TIME-PROPORTIONAL OUTPUT	TPO	4	6.9	10.2	Input condition OFF
			37	65	Input condition ON and duty specified or output limit disabled
SCALING	SCL	4	7.6	9.3	---
SCALING 2	SCL2	4	6.8	9.2	---
SCALING 3	SCL3	4	7.8	9.9	---
AVERAGE	AVG	4	22	40	Average of an operation
			212	351	Average of 64 operations

4-1-18 Subroutine Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SUBROUTINE CALL	SBS	2	0.90	2.8	---
SUBROUTINE ENTRY	SBN	2	2.8	4.1	---
SUBROUTINE RETURN	RET	1	0.43	2.0	---
MACRO	MCRO	4	16.8	21.7	---
GLOBAL SUBROUTINE RETURN	GSBS	2	0.90	2.8	---
GLOBAL SUBROUTINE CALL	GSBN	2	2.7	3.6	---
GLOBAL SUBROUTINE ENTRY	GRET	1	0.43	2.0	---

4-1-19 Interrupt Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SET INTERRUPT MASK	MSKS	3	10.6	22.1	---
READ INTERRUPT MASK	MSKR	3	9.6	14.8	---
CLEAR INTERRUPT	CLI	3	10.1	21.6	---
DISABLE INTERRUPTS	DI	1	10.3	20.4	---
ENABLE INTERRUPTS	EI	1	9.3	16.0	---

4-1-20 High-speed Counter and Pulse Output Instructions (When a Pulse I/O Module Is Connected)

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
MODE CONTROL	INI	4	---	7.4	Starting high-speed counter comparison
			---	4.0	Stopping high-speed counter comparison
			---	9.9	Changing pulse output PV
			---	8.0	Changing high-speed counter PV
			---	5.0	Changing PV of interrupt input in counter mode
			---	9.2	Stopping pulse output
			---	5.2	Stopping PWM(891) output
			---	48.2	Changes origin search/return settings

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
HIGH-SPEED COUNTER PV	PRV	4	---	7.6	Reading pulse output PV
			---	6.3	Reading high-speed counter PV
			---	3.7	Reading PV of interrupt input in counter mode
			---	7.1	Reading PV in interrupt input counter mode
			---	6.3	Reading high-speed counter status
			---	4.1	Reading PWM(891) status
			---	17.5	Reading high-speed counter range comparison results for 8 ranges
			---	34.6	Reading high-speed counter range comparison results for 32 ranges
			---	4.9	Reading frequency of high-speed counter 0
COUNTER FREQUENCY CONVERT	PRV2	4	---	5.2	---
COMPARISON TABLE LOAD	CTBL	4	---	31.7	Registering target value table and starting comparison for 1 target value
			---	1528	Registering target value table and starting comparison for 48 target values
			---	45.0	Registering range table and starting comparison for 8 ranges
			---	150.8	Registering range table and starting comparison for 32 ranges
			---	26.2	Only registering target value table for 1 target value
			---	1520	Only registering target value table for 48 target values
			---	41.8	Only registering range table for 8 ranges
			---	150.0	Only registering range table for 32 ranges
SPEED OUTPUT	SPED	4	---	23.5	Continuous mode
			---	24.6	Independent mode
SET PULSES	PULS	4	---	8.2	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
PULSE OUTPUT	PLS2	5	---	39.4	---
ACCELERATION CONTROL	ACC	4	---	35.0	Continuous mode
			---	44.6	Independent mode
ORIGIN SEARCH	ORG	3	---	28.1	Origin search
			---	23.6	Origin return
PULSE WITH VARIABLE DUTY FACTOR	PWM	4	---	7.8	---
INTERRUPT FEEDING	IFEED	4	---	42.4	---

4-1-21 Step Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
STEP DEFINE	STEP	2	8.7	10.6	Step control bit ON
			8.7	9.8	Step control bit OFF
STEP START	SNXT	2	2.2	2.8	---

4-1-22 Basic I/O Unit Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CJ2H CPU6□(-EIP)		CJ2M CPU□□		
			MIN value	MAX value	MIN value	MAX value	
I/O REFRESH	IORF	3	10.1		12.2		1-word refresh (IN) for Basic I/O Units
			10.5		13.0		1-word refresh (OUT) for Basic I/O Units
SPECIAL I/O UNIT I/O REFRESH	FIORF	2	*1		*1		---
CPU BUS I/O REFRESH	DLNK	4	234		256		Allocated 1 word
7-SEGMENT DECODER	SDEC	4	2.5		3.3		---
DIGITAL SWITCH INPUT	DSW	6	24.8		39.6		4 digits, data input value: 0
			24.8		40.2		8 digits, data input value: 00
TEN KEY INPUT	TKY	4	7.2		9.7		Data input value: 00
			6.5		8.6		Data input value: FF
HEXADECIMAL KEY INPUT	HKY	5	25.9		40.8		Data input value: 00
			25.9		41.0		Data input value: FF
MATRIX INPUT	MTR	5	25.0		38.5		Data input value: 00
			25.0		38.5		Data input value: FF
7-SEGMENT DISPLAY OUTPUT	7SEG	5	31.4		51.9		4 digits
			34.6		59.4		8 digits

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CJ2H CPU6□(-EIP)		CJ2M CPU□□		
			MIN value	MAX value	MIN value	MAX value	
ANALOG INPUT DIRECT CONVERSION	AIDC	3	25.2	30.5	27.4	31.7	Analog input number: 1, number of analog inputs used: 4
			26.9	32.1	29.1	33.5	Analog input number: 2, number of analog inputs used: 4
			32.9	38.3	35.1	39.8	Analog input number: 3, number of analog inputs used: 4
			34.5	39.8	36.8	41.9	Analog input number: 4, number of analog inputs used: 4
			28.9	34.5	31.2	35.8	Analog input number: 0, number of analog inputs used: 1
			30.8	36.4	33	38.1	Analog input number: 0, number of analog inputs used: 2
			37.2	43.3	39.5	44.5	Analog input number: 0, number of analog inputs used: 3
			39.4	45.5	41.5	46.6	Analog input number: 0, number of analog inputs used: 4
ANALOG OUTPUT DIRECT CONVERSION	AODC	3	23.1	28.9	25.4	29.7	Analog output number: 1, number of analog outputs used: 4
			23.1	28.9	25.4	29.6	Analog output number: 2, number of analog outputs used: 4
			23.1	28.9	25.3	29.7	Analog output number: 3, number of analog outputs used: 4
			23.1	28.9	25.4	29.7	Analog output number: 4, number of analog outputs used: 4
			26.6	32.6	28.9	33.3	Analog output number: 0, number of analog outputs used: 1
			32.3	38.2	34	39.1	Analog output number: 0, number of analog outputs used: 2
			38.2	43.7	40.3	45.3	Analog output number: 0, number of analog outputs used: 3
			44.1	50.0	45.8	50.6	Analog output number: 0, number of analog outputs used: 4
PCU HIGH-SPEED POSITIONING	NCDMV	4	81.7		95.3		---
PCU POSITIONING TRIGGER	NCDTR	3	22.9		25.5		---
INTELLIGENT I/O READ	IORD	4	*1		*1		---
INTELLIGENT I/O WRITE	IOWR	4	*1		*1		---

*1 Execution of the IORD, IORW, and FIORF instructions depends on the Special I/O Units for which they are being executed.

4-1-23 Serial Communications Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
PROTOCOL MACRO	PMCR	5	57.8	97.8	Direct specification
			77	132	Operand specification, sending 1 word, receiving 1 word
PROTOCOL MACRO 2	PMCR2	6	49.5	96.0	Direct specification
			69	129	Operand specification, sending 1 word, receiving 1 word
TRANSMIT	TXD	4	57.5	93.8	Sending 1 byte
			517	947	Sending 256 bytes
RECEIVE	RXD	4	79	128	Storing 1 byte
			570	1033	Storing 256 bytes
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	4	75	130	Sending 1 byte
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	4	74	128	Storing 1 byte
DIRECT TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	DTXDU	4	25.8	37.0	Sending 1 byte
			179	203	Sending 256 bytes
DIRECT RECEIVE VIA SERIAL COMMUNICATIONS UNIT	DRXDU	4	27.8	39.7	Storing 1 byte
			188	205	Storing 256 bytes
CHANGE SERIAL PORT SETUP	STUP	3	233	276	Addressed to COM port on CPU Unit

4-1-24 Network Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
NETWORK SEND	SEND	4	44.3	79.4	---
NETWORK SEND 2	SEND2	5	43.4	82.8	---
NETWORK RECEIVE	RECV	4	43.9	79.9	---
NETWORK RECEIVE 2	RECV2	5	44.5	82.8	---
DELIVER COMMAND	CMND	4	52.7	95.1	---
DELIVER COMMAND 2	CMND2	5	53.0	98.1	---
EXPLICIT MESSAGE SEND	EXPLT	4	78	134	---
EXPLICIT GET ATTRIBUTE	EGATR	4	74	127	---
EXPLICIT SET ATTRIBUTE	ESATR	3	69	117	---
EXPLICIT WORD READ	ECHRD	4	65	110	---
EXPLICIT WORD WRITE	ECHWR	4	64	110	---

4-1-25 File Memory Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
READ DATA FILE	FREAD	5	217	372	In binary
WRITE DATA FILE	FWRIT	5	216	366	In binary
WRITE TEXT FILE	TWRIT	5	205	370	---

4-1-26 Display Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
DISPLAY MESSAGE	MSG	3	6.9	10.5	Displaying message
			6.6	9.5	Deleting displayed message

4-1-27 Clock Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
CALENDAR ADD	CADD	4	15.6	22.5	---
CALENDAR SUBTRACT	CSUB	4	16.4	24.9	---
HOURS TO SECONDS	SEC	3	3.6	4.1	---
SECONDS TO HOURS	HMS	3	3.5	4.0	---
CLOCK ADJUSTMENT	DATE	2	29.6	53.2	---

4-1-28 Debugging Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
TRACE MEMORY SAMPLING	TRSM	1	8.9	12.6	Sampling 1 bit and 0 words
			31.6	33.1	Sampling 31 bits and 6 words
			38.8	39.2	Sampling 31 bits and 16 words

4-1-29 Failure Diagnosis Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
FAILURE ALARM	FAL	3	7.9	14.7	Recording errors
			14.7	22.3	Deleting errors (in order of priority)
			12.9	22.5	Deleting errors (all errors)
			117	210	Deleting errors (individually)
SEVERE FAILURE ALARM	FALS	3	---	---	---
FAILURE POINT DETECTION	FPD	4	111	188	Bit address output, time monitored
			107	202	Bit address output, first error detection
			129	242	Message characters output, time monitored
			159	244	Message characters output, first error detection

4-1-30 Other Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
SET CARRY	STC	1	0.048	0.060	---
CLEAR CARRY	CLC	1	0.048	0.060	---
SELECT EM BANK	EMBC	2	7.6	14.6	---
EXTEND MAXIMUM CYCLE TIME	WDT	2	7.6	17.1	---
SAVE CONDITION FLAGS	CCS	1	5.8	8.3	---
LOAD CONDITION FLAGS	CCL	1	6.4	9.9	---
CONVERT ADDRESS FROM CV	FRMCV	3	9.4	15.7	---
CONVERT ADDRESS TO CV	TOCV	3 to 4	10.3	18.2	---
DISABLE PERIPHERAL SERVICING	IOSP	1	---	---	---
ENABLE PERIPHERAL SERVICING	IORS	1	---	---	---

4-1-31 Block Programming Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
BLOCK PROGRAM BEGIN	BPRG	2	7.8	14.1	---
BLOCK PROGRAM END	BEND	1	8.8	13.4	---
BLOCK PROGRAM PAUSE	BPPS	2	5.4	8.4	---
BLOCK PROGRAM RESTART	BPRS	2	3.6	4.8	---
CONDITIONAL BLOCK EXIT	(Execution condition) EXIT	1	8.6	13.2	Block exited (input condition ON)
			2.0	2.6	Block not exited (input condition OFF)
CONDITIONAL BLOCK EXIT	EXIT (bit address)	2	9.8	14.8	Block exited (bit ON)
			3.6	4.2	Block not exited (bit OFF)
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT (bit address)	2	3.6	4.3	Block exited (bit OFF)
			8.9	14.9	Block not exited (bit ON)
Branching	IF (execution condition)	1	1.9	2.4	IF true (input condition ON)
			3.8	6.4	IF false (input condition OFF)
	IF (bit address)	2	3.2	4.0	IF true (bit ON)
			5.1	8.0	IF false (bit OFF)
Branching (NOT)	IF NOT (bit address)	2	5.1	8.2	IF true (bit OFF)
			3.2	4.1	IF false (bit ON)
Branching	ELSE	1	3.5	5.7	IF true
			5.3	7.3	IF false
Branching	IEND	1	5.3	8.5	IF true
			2.0	2.4	IF false

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
ONE CYCLE AND WAIT	WAIT (execution condition)	1	10.0	15.9	Do not wait (input condition ON)
			1.4	1.9	Wait (input condition OFF)
	WAIT (bit address)	2	9.2	13.5	Do not wait (bit ON)
			2.6	3.7	Wait (bit OFF)
ONE CYCLE AND WAIT (NOT)	WAIT NOT (bit address)	2	9.2	13.5	Do not wait (bit OFF)
			2.8	3.7	Wait (bit ON)
HUNDRED-MS TIMER WAIT	TIMW	3	15.6	22.9	Default setting
			16.0	23.2	Normal execution
	TIMWX	3	15.1	21.7	Default setting
			16.0	23.2	Normal execution
TEN-MS TIMER WAIT	TMHW	3	15.7	22.6	Default setting
			17.5	24.9	Normal execution
	TMHWX	3	15.2	22.1	Default setting
			16.4	23.4	Normal execution
COUNTER WAIT	CNTW	4	13.7	20.5	Default setting
			13.4	19.8	Normal execution
	CNTWX	4	13.1	19.5	Default setting
			13.5	19.7	Normal execution
Loop Control	LOOP	1	4.6	9.1	---
Loop Control	LEND (execution condition)	1	4.2	8.6	Do not loop (input condition ON)
			3.9	6.5	Loop (input condition OFF)
	LEND (bit address)	2	6.7	10.4	Do not loop (bit ON)
			6.6	8.2	Loop (bit OFF)
Loop Control (NOT)	LEND NOT (bit address)	2	6.7	10.9	Do not loop (bit OFF)
			6.6	8.2	Loop (bit ON)

4-1-32 Text String Processing Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
MOV STRING	MOV$	3	31.5	58.3	Transferring 1 character*1
CONCATENATE STRING	+$	4	56	104	1 character + 1 character*1
GET STRING LEFT	LEFT$	4	33.5	62.2	Retrieving 1 character from 2 characters*1
GET STRING RIGHT	RGHT$	4	33.4	62.1	Retrieving 1 character from 2 characters*1
GET STRING MIDDLE	MID$	5	32.3	60.8	Retrieving 1 character from 3 characters*1
FIND IN STRING	FIND$	4	30.3	56.3	Searching for 1 character from 2 characters*1

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
STRING LENGTH	LEN$	3	14.0	24.9	Detecting 1 character[*1]
REPLACE IN STRING	RPLC$	6	110	213	Replacing the first of 2 characters with 1 character[*1]
DELETE STRING	DEL$	5	45.6	86.8	Deleting the leading character of 2 characters[*1]
EXCHANGE STRING	XCHG$	3	40.3	75.4	Exchanging 1 character with 1 character[*1]
CLEAR STRING	CLR$	2	15.9	28.3	Clearing 1 character[*1]
INSERT INTO STRING	INS$	5	85	162	Inserting 1 character after the first of 2 characters[*1]
String Comparison Instructions	=$	4	27.0	50.9	Comparing 1 character with 1 character
	<>$				
	<$				
	<=$				
	>$				
	=$				

[*1] The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-1-33 Task Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
TASK ON	TKON	2	12.5	19.0	Cyclic task specified
			13.6	22.7	Extra task specified
TASK OFF	TKOF	2	240	393	Cyclic task specified
			15.5	25.8	Extra task specified

4-1-34 Model Conversion Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
BLOCK TRANSFER	XFERC	4	6.7	8.2	Transferring 1 word
			362	409	Transferring 1,000 words
SINGLE WORD DISTRIBUTE	DISTC	4	4.6	5.3	Data distribute
			6.0	7.3	Stack operation
DATA COLLECT	COLLC	4	5.3	6.5	Data distribute
			4.5	13.0	Stack operation
			5.8	6.1	Stack operation 1 word FIFO Read
			42	142	Stack operation 1,000 word FIFO Read
MOVE BIT	MOVBC	4	4.9	5.7	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
BIT COUNTER	BCNTC	4	5.5	6.4	Counting 1 word
			873	974	Counting 1,000 words

4-1-35 Special Function Block Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
GET VARIABLE ID	GETID	4	7.6	12.5	---

4-1-36 SFC Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)		Conditions
			CJ2H CPU6□(-EIP)	CJ2M CPU□□	
STEP ACTIVATE	SA	2	9.1	11.8	---
STEP DEACTIVATE	SE	2	9.1	11.9	---
READ SET TIMER	TSR	3	3.9	4.2	---
SET STEP TIMER	TSW	3	5.7	8.2	---
SFC ON	SFCON	2	14.0	20.4	---
SFC OFF	SFCOFF	2	249	402	---
SFC PAUSE WITH RESET	SFCPR	2	249	405	---
SFC PAUSE WITH NO RESET	SFCPRN	2	249	405	---

4-1-37 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			CPU Unit model	
			CJ2H-CPU6□H-EIP	CJ2M CPU□□
A	Startup time	Startup time not including I/O parameter transfer	3.3 μs	7.4 μs
B	I/O parameter transfer processing time The data type is indicated in parentheses.	1-bit (BOOL) input symbol or output symbol	0.24 μs	0.88 μs
		1-word (INT, UINT, WORD) input symbol or output symbol	0.19 μs	0.88 μs
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	0.19 μs	1.2 μs
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	0.38 μs	2.96 μs
		I/O symbols	0.114 μs	0.4 μs
C	Function block definition instruction execution time	Total instruction processing time (same as standard user program)		

Example: CJ2H-CPU67H-EIP
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 = 3.3 μs + (3 + 2) × 0.19 μs + 10 μs = 14.25 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.

Note Number of Function Block Program Steps

■ CS1-H/CJ1-H/CJ1M CPU Unit

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			CJ2 CPU Units
m	Call part	---	57 steps
n	I/O parameter transfer part The data type is shown in parentheses.	1-bit (BOOL) input symbol or output symbol	6 steps
		1-word (INT, UINT, WORD) input symbol or output symbol	6 steps
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	6 steps
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	18 steps
		I/O symbols	6 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

Note Number of Function Block Program Steps

■ CJ2H CPU Units

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			CJ2 CPU Units
m	Call part	---	52 steps
n	I/O parameter transfer part The data type is shown in parentheses.	1-bit (BOOL) input symbol or output symbol	6 steps
		1-word (INT, UINT, WORD) input symbol or output symbol	6 steps
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	6 steps
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	18 steps
		I/O symbols	6 steps
p	Number of instruction steps in function block definition	The total number of instruction steps (same as standard user program) + 11 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 = 52 + (5 + 5) × 6 steps + 100 steps + 11 steps
= 223 steps

■ CJ2M CPU Units

Follow the guidelines below for the space (number of steps) used by a function block.

Amount of UM used (number of steps)
= Number of instances × (Call part size m + I/O parameter transfer part size n × Number of parameters)

Contents			Number of steps used
m	Call part	---	15 steps
n	I/O parameter transfer part The data type is shown in parentheses.	1-bit (BOOL) input symbol or output symbol	3 steps
		1-word (INT, UINT, WORD) input symbol or output symbol	3 steps
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	3 steps
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	6 steps
		I/O symbols	9 steps

Amount of FB program area used (number of steps)
= Number of instances × (Call part size p + I/O parameter transfer part size q × Number of parameters) + Number of instruction steps in the function block definition r (See note.)

Contents			Number of steps used
p	Call part	---	37 steps
q	I/O parameter transfer part The data type is shown in parentheses.	1-bit (BOOL) input symbol or output symbol	3 steps
		1-word (INT, UINT, WORD) input symbol or output symbol	3 steps
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	3 steps
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	6 steps
		I/O symbols	9 steps
r	Number of instruction steps in function block definition	The total number of instruction steps (same as standard user program) + 11 steps.	

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).

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
Amount of UM used by one instance = 15 + (5 + 5) x 3 steps = 45 steps, amount of FB program area used = 37 + (5 + 5) x 3 steps + 100 steps + 11 steps = 178 steps.

4-2 CJ1 CPU Unit Instruction Execution Times and Number of Steps

The following table lists the execution times for all instructions that are supported by the CPU Units.

The total execution time of instructions within one whole user program is the process time for program execution when calculating the cycle time (See note.).

Note User programs are allocated tasks that can be executed within cyclic tasks and interrupt tasks that satisfy interrupt conditions.

Execution times for most instructions differ depending on the CPU Unit used and the conditions when the instruction is executed.

The execution time can also vary when the execution condition is OFF.

The following table also lists the length of each instruction in the *Length (steps)* column. The number of steps required in the user program area for each instructions depends on 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. Most instructions are supported in differentiated form (indicated with ↑, ↓, @, and %). Specifying differentiation will increase the execution times by the following amounts. (Unit: µs)

Symbol	CJ1-H			CJ1M	CJ1
	CPU6□H-R	CPU6□H	CPU4□H	CPU□□	CPU4□
↑ or ↓	+0.24	+0.24	+0.32	+0.5	+0.45
@ or %	+0.24	+0.24	+0.32	+0.5	+0.33

2. Use the following times as guidelines when instructions are not executed. (Unit: µs)

CJ1-H			CJ1M	CJ1
CPU6□H-R	CPU6□H	CPU4□H	CPU□□	CPU4□
0.016 to 0.096	0.018 to 0.108	0.02 to 0.12	0.05 to 0.30	0.12 to 0.72

4-2-1 Sequence Input Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□H-R	CJ1H CPU6□H	CJ1G CPU4□H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
LOAD	LD	1	0.016	0.02	0.04	0.10	0.10	0.08	---
	!LD	2	21.16	21.16	21.20	24.20	28.17	21.24	---
LOAD NOT	LD NOT	1	0.016	0.02	0.04	0.10	0.10	0.08	---
	!LD NOT	2	21.16	21.16	21.20	24.20	28.17	21.24	---
AND	AND	1	0.016	0.02	0.04	0.10	0.10	0.08	---
	!AND	2	21.16	21.16	21.20	24.20	28.17	21.24	---
AND NOT	AND NOT	1	0.016	0.02	0.04	0.10	0.10	0.08	---
	!AND NOT	2	21.16	21.16	21.20	24.20	28.17	21.24	---
OR	OR	1	0.016	0.02	0.04	0.10	0.10	0.08	---
	!OR	2	21.16	21.16	21.20	24.20	28.17	21.24	---
OR NOT	OR NOT	1	0.016	0.02	0.04	0.10	0.10	0.08	---
	!OR NOT	2	21.16	21.16	21.20	24.20	28.17	21.24	---
AND LOAD	AND LD	1	0.016	0.02	0.04	0.05	0.05	0.08	---
OR LOAD	OR LD	1	0.016	0.02	0.04	0.05	0.05	0.08	---
NOT	NOT	1	0.016	0.02	0.04	0.05	0.05	0.08	---
CONDITION ON	UP	3	0.24	0.3	0.42	0.50	0.50	0.54	---
CONDITION OFF	DOWN	4	0.24	0.3	0.42	0.50	0.50	0.54	---
LOAD BIT TEST	LD TST	4	0.11	0.14	0.24	0.35	0.35	0.37	---
LOAD BIT TEST NOT	LD TSTN	4	0.11	0.14	0.24	0.35	0.35	0.37	---
AND BIT TEST NOT	AND TSTN	4	0.11	0.14	0.24	0.35	0.35	0.37	---
OR BIT TEST	OR TST	4	0.11	0.14	0.24	0.35	0.35	0.37	---
OR BIT TEST NOT	OR TSTN	4	0.11	0.14	0.24	0.35	0.35	0.37	---

4-2-2 Sequence Output Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
OUTPUT	OUT	1	0.016	0.02	0.04	0.35	0.35	0.21	---
	!OUT	2	21.39	21.39	21.41	23.42	28.95	21.58	---
OUTPUT NOT	OUT NOT	1	0.016	0.02	0.04	0.35	0.35	0.21	---
	!OUT NOT	2	21.39	21.39	21.41	23.42	28.95	21.58	---
KEEP	KEEP	1	0.048	0.06	0.08	0.40	0.40	0.29	---
	!KEEP	1	21.42	21.43	21.45	23.47	29.0	21.66	
DIFFERENTIATE UP	DIFU	2	0.21	0.24	0.40	0.50	0.50	0.54	---
DIFFERENTIATE DOWN	DIFD	2	0.21	0.24	0.40	0.50	0.50	0.54	---
SET	SET	1	0.016	0.02	0.06	0.30	0.30	0.21	---
	!SET	2	21.39	21.39	21.43	23.47	28.90	21.58	---
RESET	RSET	1	0.016	0.02	0.06	0.30	0.30	0.21	---
	!RSET	2	21.39	21.39	21.43	23.47	28.90	21.58	---

Instruction	Mnemonic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
MULTIPLE BIT SET	SETA	4	5.8	5.8	6.1	11.8	11.8	7.8	With 1-bit set
			25.7	25.7	27.2	64.1	64.1	38.8	With 1,000-bit set
MULTIPLE BIT RESET	RSTA	4	5.7	5.7	6.1	11.8	11.8	7.8	With 1-bit reset
			25.8	25.8	27.1	64.0	64.0	38.8	With 1,000-bit reset
SINGLE BIT SET	SETB	2	0.19	0.24	0.34	0.5	0.5	---	---
	!SETB	3	21.63	21.48	21.88	23.81	23.81	---	---
SINGLE BIT RESET	RSTB	2	0.19	0.24	0.34	0.5	0.5	---	---
	!RSTB	3	21.64	21.48	21.88	23.81	23.81	---	---
SINGLE BIT OUTPUT	OUTB	2	0.19	0.22	0.32	0.45	0.45	---	---
	!OUTB	3	21.61	21.64	21.84	23.67	23.67	---	---

4-2-3 Sequence Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
END	END	1	5.5	5.5	6.0	7.9	7.9	4.0	---
NO OPERATION	NOP	1	0.016	0.02	0.04	0.05	0.05	0.12	---
INTERLOCK	IL	1	0.048	0.06	0.06	0.15	0.15	0.12	---
INTERLOCK CLEAR	ILC	1	0.048	0.06	0.06	0.15	0.15	0.12	---
MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	3	7.5	7.5	7.9	13.3	14.6	---	Interlock condition not met (input condition ON)
			8.9	8.9	9.7	16.6	18.3	---	Interlock condition met (input condition OFF)
			6.1	6.1	6.5	10.3	11.7	---	Interlock condition met again during interlock (input condition OFF)
MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	3	7.5	7.5	7.9	13.3	14.6	---	Interlock condition not met (input condition ON)
			8.9	8.9	9.7	16.6	18.3	---	Interlock condition met (input condition OFF)
			6.1	6.1	6.5	10.3	11.7	---	Interlock condition met again during interlock (input condition OFF)
MULTI-INTERLOCK CLEAR	MILC	2	5.0	5.0	5.6	8.3	12.5	---	Not during interlock
			5.7	5.7	6.2	9.6	14.2	---	During interlock
JUMP	JMP	2	0.31	0.38	0.48	0.95	0.95	8.1	---
JUMP END	JME	2	---	---	---	---	---	---	---
CONDITIONAL JUMP	CJP	2	0.31	0.38	0.48	0.95	0.95	7.4	Jump condition met (input condition ON)

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
CONDITIONAL JUMP NOT	CJPN	2	0.31	0.38	0.48	0.95	0.95	8.5	Jump condition met (input condition OFF)
MULTIPLE JUMP	JMP0	1	0.048	0.06	0.06	0.15	0.15	0.12	---
MULTIPLE JUMP END	JME0	1	0.048	0.06	0.06	0.15	0.15	0.12	---
FOR LOOP	FOR	2	0.18	0.21	0.21	1.00	1.00	0.21	Designating a constant
BREAK LOOP	BREAK	1	0.048	0.12	0.12	0.15	0.15	0.12	---
NEXT LOOP	NEXT	1	0.14	0.18	0.18	0.45	0.45	0.18	When loop is continued
			0.18	0.22	0.22	0.55	0.55	0.22	When loop is ended

4-2-4 Timer and Counter Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
HUNDRED-MS TIMER	TIM	3	0.45	0.56	0.88	1.30	1.30	0.42	---
	TIMX		0.45						
TEN-MS TIMER	TIMH	3	0.70	0.88	1.14	1.80	1.80	0.42	---
	TIMHX		0.46	0.56	0.88	1.30	1.30	0.42	
ONE-MS TIMER	TMHH	3	0.69	0.86	1.12	1.75	1.75	0.42	---
	TMHHX		0.46	0.56	0.88	1.30	1.30	0.42	
TENTH-MS TIMER	TIMU	3	0.45	---	---	---	---	---	---
	TIMUX		0.45						
HUNDREDTH-MS TIMER	TMUH	3	0.45	---	---	---	---	---	---
	TMUHX		0.45						
ACCUMULATIVE TIMER	TTIM	3	16.1	16.1	17.0	27.4	30.9	21.4	---
			10.9	10.9	11.4	19.0	21.2	14.8	When resetting
			8.5	8.5	8.7	15.0	16.6	10.7	When interlocking
	TTIMX	3	16.1	16.1	17.0	27.4	---	21.4	---
			10.9	10.9	11.4	19.0	---	14.8	When resetting
			8.5	8.5	8.7	15.0	---	10.7	When interlocking
LONG TIMER	TIML	4 to 5	7.6	7.6	10.0	16.3	17.2	12.8	---
			6.2	6.2	6.5	13.8	15.3	7.8	When interlocking
	TIMLX	4 to 5	7.6	7.6	10.0	16.3	---	12.8	---
			6.2	6.2	6.5	13.8	---	7.8	When interlocking
MULTI-OUTPUT TIMER	MTIM	4	20.9	20.9	23.3	38.55	43.3	26.0	---
			5.6	5.6	5.8	12.9	13.73	7.8	When resetting
	MTIMX	4	20.9	20.9	23.3	38.55	---	26.0	---
			5.6	5.6	5.8	12.9	---	7.8	When resetting
COUNTER	CNT	3	0.51	0.56	0.88	1.30	1.30	0.42	---
	CNTX		0.51				---		
REVERSIBLE COUNTER	CNTR	3	16.9	16.9	19.0	31.8	27.2	20.9	---
	CNTRX		16.9				---		
RESET TIMER/COUNTER	CNR	3	9.9	9.9	10.6	14.7	17.93	13.9	When resetting 1 word
			4160	4160	4160	6210	6300	5420	When resetting 1,000 words
	CNRX	3	9.9	9.9	10.6	14.7	17.93	13.9	When resetting 1 word
			4160	4160	4160	6210	6300	5420	When resetting 1,000 words

4-2-5 Comparison Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
Input Comparison Instructions (unsigned)	=	4	0.08	0.10	0.16	0.35	0.35	0.37	---
	<>								
	<								
	<=								
	>								
	>=								
Input Comparison Instructions (double, unsigned)	=L	4 to 7	0.08	0.10	0.16	0.35	0.35	0.54	---
	<>L								
	<L								
	<=L								
	>L								
	>=L								
Input Comparison Instructions (signed)	=S	4	0.08	0.10	0.16	0.35	0.35	6.50	---
	<>S								
	<S								
	<=S								
	>S								
	>=S								
Input Comparison Instructions (double, signed)	=SL	4 to 7	0.08	0.10	0.16	0.35	0.35	6.50	---
	<>SL								
	<SL								
	<=SL								
	>SL								
	>=SL								
Time Comparison Instructions	=DT	4	25.2	25.2	36.4	45.6	40.6	---	---
	<>DT								
	<DT								
	<=DT								
	>DT								
	>=DT								
COMPARE	CMP	3	0.032	0.04	0.04	0.10	0.10	0.29	---
	!CMP	7	42.2	42.2	42.2	45.3	45.3	42.7	---
DOUBLE COMPARE	CMPL	3 to 5	0.064	0.08	0.08	0.50	0.50	0.46	---
SIGNED BINARY COMPARE	CPS	3	0.064	0.08	0.08	0.30	0.30	6.50	---
	!CPS	7	36.0	36.0	36.0	45.5	45.5	48.9	---
DOUBLE SIGNED BINARY COMPARE	CPSL	3 to 5	0.064	0.08	0.08	0.50	0.50	6.50	---
TABLE COMPARE	TCMP	4	14.0	14.0	15.2	29.77	32.13	21.9	---
MULTIPLE COMPARE	MCMP	4	20.5	20.5	22.8	45.80	48.67	31.2	---
UNSIGNED BLOCK COMPARE	BCMP	4	21.5	21.5	23.7	47.93	51.67	32.6	---

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
EXPANDED BLOCK COMPARE	BCMP2	4	8.4	---	---	13.20	19.33	---	Number of data words: 1
			313.0	---	---	650.0	754.67	---	Number of data words: 255
AREA RANGE COMPARE	ZCP	3	5.3	5.3	5.4	11.53	12.43	---	---
DOUBLE AREA RANGE COMPARE	ZCPL	3 to 5	5.5	5.5	6.7	11.28	11.90	---	---

4-2-6 Data Movement Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
MOVE	MOV	3	0.14	0.18	0.20	0.30	0.30	0.29	---
	!MOV	7	21.52	21.56	21.60	35.4	43.3	42.65	---
DOUBLE MOVE	MOVL	3 to 4	0.26	0.32	0.34	0.60	0.60	0.50	---
MOVE NOT	MVN	3	0.14	0.18	0.20	0.35	0.35	0.29	---
DOUBLE MOVE NOT	MVNL	3 to 4	0.26	0.32	0.34	0.60	0.60	0.50	---
MOVE BIT	MOVB	4	0.19	0.24	0.34	0.50	0.50	7.5	---
MOVE DIGIT	MOVD	4	0.19	0.24	0.34	0.50	0.50	7.3	---
MULTIPLE BIT TRANSFER	XFRB	4	10.1	10.1	10.8	20.9	22.1	13.6	Transferring 1 bit
			186.4	186.4	189.8	253.3	329.7	269.2	Transferring 255 bits
BLOCK TRANSFER	XFER	4	0.29	0.36	0.44	0.8	0.8	11.2	Transferring 1 word
			240.1	300.1	380.1	650.2	650.2	633.5	Transferring 1,000 words
BLOCK SET	BSET	4	0.21	0.26	0.28	0.55	0.55	8.5	Setting 1 word
			142.2	200.1	220.1	400.2	400.2	278.3	Setting 1,000 words
DATA EXCHANGE	XCHG	3	0.32	0.40	0.56	0.80	0.80	0.7	---
DOUBLE DATA EXCHANGE	XCGL	3	0.61	0.76	1.04	1.5	1.5	1.3	---
SINGLE WORD DISTRIBUTE	DIST	4	5.1	5.1	5.4	6.6	12.47	7.0	---
DATA COLLECT	COLL	4	5.1	5.1	5.3	6.5	12.77	7.1	---
MOVE TO REGISTER	MOVR	3	0.064	0.08	0.08	0.60	0.60	0.50	---
MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	3	0.064	0.42	0.50	0.60	0.60	0.50	---

4-2-7 Data Shift Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
SHIFT REGISTER	SFT	3	7.4	7.4	10.4	11.9	15.3	10.4	Shifting 1 word
			433	433	488	1390	1430	763	Shifting 1,000 words
REVERSIBLE SHIFT REGISTER	SFTR	4	6.9	6.9	7.2	11.4	15.5	9.6	Shifting 1 word
			615.3	615.3	680.2	1430	1550	859.6	Shifting 1,000 words
ASYNCHRO-NOUS SHIFT REGISTER	ASFT	4	6.2	6.2	6.4	13.4	14.2	7.7	Shifting 1 word
			1220	1220	1220	2750	2990	2010	Shifting 1,000 words[*1]
WORD SHIFT	WSFT	4	4.5	4.5	4.7	9.6	12.3	7.8	Shifting 1 word
			171.5	171.5	171.7	928.0	933.3	781.7	Shifting 1,000 words
ARITHMETIC SHIFT LEFT	ASL	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE SHIFT LEFT	ASLL	2	0.32	0.40	0.56	0.80	0.80	0.67	---
ARITHMETIC SHIFT RIGHT	ASR	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE SHIFT RIGHT	ASRL	2	0.32	0.40	0.56	0.80	0.80	0.67	---
ROTATE LEFT	ROL	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE ROTATE LEFT	ROLL	2	0.32	0.40	0.56	0.80	0.80	0.67	---
ROTATE LEFT WITHOUT CARRY	RLNC	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	2	0.32	0.40	0.56	0.80	0.80	0.67	---
ROTATE RIGHT	ROR	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE ROTATE RIGHT	RORL	2	0.32	0.40	0.56	0.80	0.80	0.67	---
ROTATE RIGHT WITHOUT CARRY	RRNC	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	2	0.32	0.40	0.56	0.80	0.80	0.67	---
ONE DIGIT SHIFT LEFT	SLD	3	5.9	5.9	6.1	7.6	12.95	8.2	Shifting 1 word
			561.1	561.1	626.3	1150	1270	760.7	Shifting 1,000 words
ONE DIGIT SHIFT RIGHT	SRD	3	6.9	6.9	7.1	8.6	15.00	8.7	Shifting 1 word
			760.5	760.5	895.5	1720	1820	1070	Shifting 1,000 words
SHIFT N-BIT DATA LEFT	NSFL	4	7.5	7.5	8.3	14.8	16.0	10.5	Shifting 1 bit
			34.5	40.3	45.4	86.7	91.3	55.5	Shifting 1,000 bits
SHIFT N-BIT DATA RIGHT	NSFR	4	7.5	7.5	8.3	14.7	15.9	10.5	Shifting 1 bit
			48.2	50.5	55.3	114.1	119.6	69.3	Shifting 1,000 bits
SHIFT N-BITS LEFT	NASL	3	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE SHIFT N-BITS LEFT	NSLL	3	0.32	0.40	0.56	0.80	0.80	0.67	---

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
SHIFT N-BITS RIGHT	NASR	3	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	3	0.32	0.40	0.56	0.80	0.80	0.67	---

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-2-8 Increment/Decrement Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
INCREMENT BINARY	++	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE INCREMENT BINARY	++L	2	0.18	0.40	0.56	0.80	0.80	0.67	---
DECREMENT BINARY	– –	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE DECREMENT BINARY	– –L	2	0.18	0.40	0.56	0.80	0.80	0.67	---
INCREMENT BCD	++B	2	5.7	6.4	4.5	12.3	14.7	7.4	---
DOUBLE INCREMENT BCD	++BL	2	5.6	5.6	4.9	9.24	10.8	6.1	---
DECREMENT BCD	– –B	2	5.7	6.3	4.6	11.9	14.9	7.2	---
DOUBLE DECREMENT BCD	– –BL	2	5.3	5.3	4.7	9.0	10.7	7.1	---

4-2-9 Symbol Math Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11 /21	CJ1G CPU4□	
SIGNED BINARY ADD WITHOUT CARRY	+	4	0.18	0.18	0.20	0.30	0.30	0.37	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	4 to 6	0.18	0.32	0.34	0.60	0.60	0.54	---
SIGNED BINARY ADD WITH CARRY	+C	4	0.18	0.18	0.20	0.40	0.40	0.37	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	4 to 6	0.18	0.32	0.34	0.60	0.60	0.54	---
BCD ADD WITHOUT CARRY	+B	4	7.6	8.2	8.4	18.9	21.5	14.0	---
DOUBLE BCD ADD WITHOUT CARRY	+BL	4 to 6	9.2	13.3	14.5	24.4	27.7	19.0	---
BCD ADD WITH CARRY	+BC	4	8.0	8.9	9.1	19.7	22.6	14.5	---
DOUBLE BCD ADD WITH CARRY	+BCL	4 to 6	9.6	13.8	15.0	25.2	28.8	19.6	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	4	0.18	0.18	0.20	0.3	0.3	0.37	---

Instruction	Mne-monic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11 /21	CJ1G CPU4□	
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	4 to 6	0.18	0.32	0.34	0.60	0.60	0.54	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	4	0.18	0.18	0.20	0.40	0.40	0.37	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	4 to 6	0.18	0.32	0.34	0.60	0.60	0.54	---
BCD SUBTRACT WITHOUT CARRY	–B	4	7.4	8.0	8.2	18.1	20.5	13.1	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	4 to 6	8.9	12.8	14.0	23.2	26.7	18.2	---
BCD SUBTRACT WITH CARRY	–BC	4	7.9	8.5	8.6	19.1	21.6	13.8	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	4 to 6	9.4	13.4	14.7	24.3	27.7	18.8	---
SIGNED BINARY MULTIPLY	*	4	0.26	0.38	0.40	0.65	0.65	0.58	---
DOUBLE SIGNED BINARY MULTIPLY	*L	4 to 6	5.93	7.23	8.45	13.17	15.0	11.19	---
UNSIGNED BINARY MULTIPLY	*U	4	0.26	0.38	0.40	0.75	0.75	0.58	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	4 to 6	5.9	7.1	8.3	13.30	15.2	10.63	---
BCD MULTIPLY	*B	4	8.3	9.0	9.2	17.5	19.7	12.8	---
DOUBLE BCD MULTIPLY	*BL	4 to 6	12.8	23.0	24.2	36.3	45.7	35.2	---
SIGNED BINARY DIVIDE	/	4	0.29	0.40	0.42	0.70	0.70	0.83	---
DOUBLE SIGNED BINARY DIVIDE	/L	4 to 6	7.2	7.2	8.4	13.7	15.5	9.8	---
UNSIGNED BINARY DIVIDE	/U	4	0.29	0.40	0.42	0.8	0.8	0.83	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	4 to 6	6.9	6.9	8.1	12.8	14.7	9.1	---
BCD DIVIDE	/B	4	8.6	8.6	8.8	19.3	22.8	15.9	---
DOUBLE BCD DIVIDE	/BL	4 to 6	13.1	17.7	18.9	27.1	34.7	26.2	---

4-2-10 Conversion Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
BCD TO BINARY	BIN	3	0.18	0.22	0.24	0.40	0.40	0.29	---
DOUBLE BCD TO DOUBLE BINARY	BINL	3 to 4	6.1	6.5	6.8	12.3	13.7	9.1	---
BINARY TO BCD	BCD	3	0.19	0.24	0.26	7.62	9.78	8.3	---
DOUBLE BINARY TO DOUBLE BCD	BCDL	3 to 4	6.7	6.7	7.0	10.6	12.8	9.2	---
2'S COMPLEMENT	NEG	3	0.14	0.18	0.20	0.35	0.35	0.29	---

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
DOUBLE 2’S COMPLEMENT	NEGL	3 to 4	0.26	0.32	0.34	0.60	0.60	0.50	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN	3	0.26	0.32	0.34	0.60	0.60	0.50	---
DATA DECODER	MLPX	4	0.32	0.32	0.42	0.85	0.85	8.8	Decoding 1 digit (4 to 16)
			0.98	0.98	1.20	1.60	1.60	12.8	Decoding 4 digits (4 to 16)
			3.30	3.30	4.00	4.70	4.70	20.3	Decoding 1 digit (8 to 256)
			6.50	6.50	7.90	8.70	8.70	33.4	Decoding 4 digits (8 to 256)
DATA ENCODER	DMPX	4	7.5	7.5	7.9	9.4	13.9	10.4	Encoding 1 digit (16 to 4)
			49.6	49.6	50.2	57.3	71.73	59.1	Encoding 4 digits (16 to 4)
			18.2	18.2	18.6	56.8	82.7	23.6	Encoding 1 digit (256 to 8)
			55.1	55.1	57.4	100.0	150.7	92.5	Encoding 2 digits (256 to 8)
ASCII CONVERT	ASC	4	6.8	6.8	7.1	8.3	14.6	9.7	Converting 1 digit into ASCII
			9.0	11.2	11.7	19.1	21.8	15.1	Converting 4 digits into ASCII
ASCII TO HEX	HEX	4	7.1	7.1	7.4	12.1	15.6	10.1	Converting 1 digit
COLUMN TO LINE	LINE	4	16.6	19.0	23.1	37.0	40.3	29.1	---
LINE TO COLUMN	COLM	4	18.4	23.2	27.5	45.7	48.2	37.3	---
SIGNED BCD TO BINARY	BINS	4	6.8	8.0	8.3	16.2	17.0	12.1	Data format setting No. 0
			6.8	8.0	8.3	16.2	17.1	12.1	Data format setting No. 1
			7.1	8.3	8.6	16.5	17.7	12.7	Data format setting No. 2
			7.4	8.5	8.8	16.5	17.6	13.0	Data format setting No. 3
DOUBLE SIGNED BCD TO BINARY	BISL	4 to 5	6.9	9.2	9.6	18.4	19.6	13.6	Data format setting No. 0
			7.0	9.2	9.6	18.5	19.8	13.7	Data format setting No. 1
			7.3	9.5	9.9	18.6	20.1	14.2	Data format setting No. 2
			7.6	9.6	10.0	18.7	20.1	14.4	Data format setting No. 3
SIGNED BINARY TO BCD	BCDS	4	6.6	6.6	6.9	13.5	16.4	10.6	Data format setting No. 0
			6.7	6.7	7.0	13.8	16.7	10.8	Data format setting No. 1
			6.8	6.8	7.1	13.9	16.8	10.9	Data format setting No. 2
			7.1	7.2	7.5	14.0	17.1	11.5	Data format setting No. 3

Instruction	Mne-monic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
DOUBLE SIGNED BINARY TO BCD	BDSL	4 to 5	7.6	8.1	8.4	11.4	12.5	11.6	Data format setting No. 0
			6.7	8.2	8.6	11.7	12.73	11.8	Data format setting No. 1
			6.7	8.3	8.7	11.8	12.8	12.0	Data format setting No. 2
			6.9	8.8	9.2	11.9	13.0	12.5	Data format setting No. 3
GRAY CODE CONVERSION	GRY	4	46.9	46.9	72.1	80.0	71.2	---	8-bit binary
			49.6	49.6	75.2	83.0	75.6	---	8-bit BCD
			57.7	57.7	87.7	95.9	86.4	---	8-bit angle
			61.8	61.8	96.7	104.5	91.6	---	15-bit binary
			64.5	64.5	99.6	107.5	96.1	---	15-bit BCD
			72.8	72.8	112.4	120.4	107.3	---	15-bit angle
			52.3	52.3	87.2	88.7	82.4	---	360° binary
			55.1	55.1	90.4	91.7	86.8	---	360° BCD
			64.8	64.8	98.5	107.3	98.1	---	360° angle
FOUR-DIGIT NUMBER TO ASCII	STR4	3	13.79	13.79	20.24	22.16	19.88	---	---
EIGHT-DIGIT NUMBER TO ASCII	STR8	3 to 4	18.82	18.82	27.44	29.55	26.70	---	---
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	3	30.54	30.54	44.41	48.16	44.10	---	---
ASCII TO FOUR-DIGIT NUMBER	NUM4	3 to 4	18.46	18.46	27.27	29.13	26.88	---	---
ASCII TO EIGHT-DIGIT NUMBER	NUM8	3	27.27	27.27	40.29	42.69	39.71	---	---
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	3	52.31	52.31	78.25	82.21	74.23	---	---

4-2-11 Logic Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
LOGICAL AND	ANDW	4	0.14	0.18	0.20	0.30	0.30	0.37	---
DOUBLE LOGICAL AND	ANDL	4 to 6	0.26	0.32	0.34	0.60	0.60	0.54	---
LOGICAL OR	ORW	4	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE LOGICAL OR	ORWL	4 to 6	0.26	0.32	0.34	0.60	0.60	0.54	---
EXCLUSIVE OR	XORW	4	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE EXCLUSIVE OR	XORL	4 to 6	0.26	0.32	0.34	0.60	0.60	0.54	---
EXCLUSIVE NOR	XNRW	4	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE EXCLUSIVE NOR	XNRL	4 to 6	0.26	0.32	0.34	0.60	0.60	0.54	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
COMPLEMENT	COM	2	0.18	0.22	0.32	0.45	0.45	0.37	---
DOUBLE COMPLEMENT	COML	2	0.32	0.40	0.56	0.80	0.80	0.67	---

4-2-12 Special Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
BINARY ROOT	ROTB	3	49.6	49.6	50.0	56.5	82.7	530.7	---
BCD SQUARE ROOT	ROOT	3	13.7	13.7	13.9	59.3	88.4	514.5	---
ARITHMETIC PROCESS	APR	4	6.7	6.7	6.9	14.0	15.0	32.3	Designating SIN and COS
			17.2	17.2	18.4	32.2	37.9	78.3	Designating line-segment approximation (with two coordinates in the line-segment data table)
FLOATING POINT DIVIDE	FDIV	4	116.6	116.6	176.6	246.0	154.7	176.6	---
BIT COUNTER	BCNT	4	0.24	0.3	0.38	0.65	0.65	22.1	Counting 1 word

4-2-13 Floating-point Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
FLOATING TO 16-BIT	FIX	3 to 4	0.13	10.6	10.8	16.2	19.5	14.5	---
FLOATING TO 32-BIT	FIXL	3 to 4	0.13	10.8	11.0	16.6	21.7	14.6	---
16-BIT TO FLOATING	FLT	3 to 4	0.13	8.3	8.5	12.2	14.6	11.1	---
32-BIT TO FLOATING	FLTL	3 to 4	0.13	8.3	8.5	14.0	15.8	10.8	---
FLOATING-POINT ADD	+F	4 to 6	0.24	8.0	9.2	13.3	15.7	10.2	---
FLOATING-POINT SUBTRACT	–F	4 to 6	0.24	8.0	9.2	13.3	15.8	10.3	---
FLOATING-POINT DIVIDE	/F	4 to 6	0.4	8.7	9.9	14.0	17.6	12.0	---
FLOATING-POINT MULTIPLY	*F	4 to 6	0.24	8.0	9.2	13.2	15.8	10.5	---
DEGREES TO RADIANS	RAD	3 to 4	8.1	10.1	10.2	15.9	20.6	14.9	---
RADIANS TO DEGREES	DEG	3 to 4	8.0	9.9	10.1	15.7	20.4	14.8	---
SINE	SIN	3 to 4	42.0	42.0	42.2	47.9	70.9	61.1	---
HIGH-SPEED SINE	SINQ	8 to 9	0.59	---	---	---	---	---	---

Instruction	Mne-monic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
COSINE	COS	3 to 4	31.5	31.5	31.8	41.8	51.0	44.1	---
HIGH-SPEED COSINE	COSQ	8 to 9	0.59	---	---	---	---	---	---
TANGENT	TAN	3 to 4	16.3	16.3	16.6	20.8	27.6	22.6	---
HIGH-SPEED TANGENT	TANQ	15 to 16	1.18	---	---	---	---	---	---
ARC SINE	ASIN	3 to 4	17.6	17.6	17.9	80.3	122.9	24.1	---
ARC COSINE	ACOS	3 to 4	20.4	20.4	20.7	25.3	33.5	28.0	---
ARC TANGENT	ATAN	3 to 4	16.1	16.1	16.4	45.9	68.9	16.4	---
SQUARE ROOT	SQRT	3 to 4	0.42	19.0	19.3	26.2	33.2	28.1	---
EXPONENT	EXP	3 to 4	65.9	65.9	66.2	68.8	108.2	96.7	---
LOGARITHM	LOG	3 to 4	12.8	12.8	13.1	69.4	103.7	17.4	---
EXPONENTIAL POWER	PWR	4 to 6	125.4	125.4	126.0	134.0	201.0	181.7	---
Floating Symbol Comparison	=F	4 to 6	0.13	6.6	8.3	12.6	15.37	---	---
	<>F								
	<F								
	<=F								
	>F								
	>=F								
FLOATING-POINT TO ASCII	FSTR	4 to 5	48.5	48.5	48.9	58.4	85.7	---	---
ASCII TO FLOATING-POINT	FVAL	3	21.1	21.1	21.3	31.1	43.73	---	---
MOVE FLOAT-ING-POINT (SIN-GLE)	MOVF	3 to 4	0.18	---	---	---	---	---	---

4-2-14 Double-precision Floating-point Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (μs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
DOUBLE SYMBOL COMPARISON	=D	4	8.5	8.5	10.3	16.2	19.9	---	---
	<>D								
	<D								
	<=D								
	>D								
	>=D								
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	3	11.0	11.7	12.1	16.1	21.6	---	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	3	10.2	11.6	12.1	16.4	21.7	---	---
16-BIT BINARY TO DOUBLE FLOATING	DBL	3	9.9	9.9	10.0	14.3	16.5	---	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	3	9.8	9.8	10.0	16.0	17.7	---	---
DOUBLE FLOATING-POINT ADD	+D	4	11.2	11.2	11.9	18.3	23.6	---	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
DOUBLE FLOATING-POINT SUBTRACT	–D	4	11.2	11.2	11.9	18.3	23.6	---	---
DOUBLE FLOATING-POINT MULTIPLY	*D	4	12.0	12.0	12.7	19.0	25.0	---	---
DOUBLE FLOATING-POINT DIVIDE	/D	4	23.5	23.5	24.2	30.5	44.3	---	---
DOUBLE DEGREES TO RADIANS	RADD	3	11.5	27.4	27.8	32.7	49.1	---	---
DOUBLE RADIANS TO DEGREES	DEGD	3	11.2	11.2	11.9	33.5	48.4	---	---
DOUBLE SINE	SIND	3	45.4	45.4	45.8	67.9	76.7	---	---
DOUBLE COSINE	COSD	3	43.0	43.0	43.4	70.9	72.3	---	---
DOUBLE TANGENT	TAND	3	19.8	20.1	20.5	97.9	157.0	---	---
DOUBLE ARC SINE	ASIND	3	21.5	21.5	21.9	32.3	37.3	---	---
DOUBLE ARC COSINE	ACOSD	3	24.7	24.7	25.1	29.9	42.5	---	---
DOUBLE ARC TANGENT	ATAND	3	19.3	19.3	19.7	24.0	34.4	---	---
DOUBLE SQUARE ROOT	SQRTD	3	47.4	47.4	47.9	52.9	81.9	---	---
DOUBLE EXPONENT	EXPD	3	121.0	121.0	121.4	126.3	201.3	---	---
DOUBLE LOGARITHM	LOGD	3	16.0	16.0	16.4	21.6	29.3	---	---
DOUBLE EXPONENTIAL POWER	PWRD	4	223.9	223.9	224.2	232.3	373.4	---	---

4-2-15 Table Data Processing Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
SET STACK	SSET	3	8.0	8.0	8.3	14.2	20.3	8.5	Designating 5 words in stack area
			231.6	231.6	251.8	426.5	435.3	276.8	Designating 1,000 words in stack area
PUSH ONTO STACK	PUSH	3	6.5	6.5	8.6	15.7	16.4	9.1	---
FIRST IN FIRST OUT	FIFO	3	6.9	6.9	8.9	15.8	16.8	10.6	Designating 5 words in stack area
			352.6	352.6	434.3	728.0	732.0	1130	Designating 1,000 words in stack area
LAST IN FIRST OUT	LIFO	3	7.0	7.0	9.0	16.6	17.2	9.9	---
DIMENSION RECORD TABLE	DIM	5	15.2	15.2	21.6	27.8	27.1	142.1	---

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
SET RECORD LOCATION	SETR	4	5.4	5.4	5.9	12.8	13.2	7.0	---
GET RECORD NUMBER	GETR	4	7.8	7.8	8.4	16.1	18.3	11.0	---
DATA SEARCH	SRCH	4	15.5	15.5	19.5	29.1	26.4	19.5	Searching for 1 word
			2420	2420	3340	4410	3600	3340	Searching for 1,000 words[*1]
SWAP BYTES	SWAP	3	12.2	12.2	13.6	21.0	18.4	13.6	Swapping 1 word
			1940	1940	2820	3650	3150	2820	Swapping 1,000 words[*1]
FIND MAXIMUM	MAX	4 to 5	19.2	19.2	24.9	35.3	32.0	24.9	Number of values being searched: 1
			2390	2390	3360	4390	3570	3360	Number of values being searched: 1,000[*1]
FIND MINIMUM	MIN	4 to 5	19.2	19.2	25.3	35.4	31.9	25.3	Number of values being searched: 1
			2390	2390	3330	4390	3580	3330	Number of values being searched: 1,000[*1]
SUM	SUM	4 to 5	28.2	28.2	38.5	49.5	44.1	38.3	Adding 1 word
			1420	1420	1950	2330	2110	1950	Adding 1,000 words[*1]
FRAME CHECKSUM	FCS	4 to 5	20.0	20.0	28.3	34.8	31.5	28.3	For 1-word table length
			1650	1650	2480	3110	2770	2480	For 1,000-word table length[*1]
STACK SIZE READ	SNUM	3	6.0	6.0	6.3	12.1	13.7	---	---
STACK DATA READ	SREAD	4	8.0	8.0	8.4	18.1	20.6	---	---
STACK DATA OVERWRITE	SWRIT	4	7.2	7.2	7.6	16.9	18.8	---	---
STACK DATA INSERT	SINS	4	7.8	7.8	9.9	18.2	20.5	---	---
			354.0	354.0	434.8	730.7	732.0	---	For 1,000-word table
STACK DATA DELETE	SDEL	4	8.6	8.6	10.6	19.3	22.0	---	---
			354.0	354.0	436.0	732.0	744.0	---	For 1,000-word table

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-2-16 Data Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□H-R	CJ1H CPU6□H	CJ1G CPU4□H	CJ1M CPU12/13/22/23	CJ1M CPU11/21	CJ1G CPU4□	
PID CONTROL	PID	4	436.2	436.2	678.2	612.0	552.6	678.2	First execution
			332.3	332.3	474.9	609.3	548.0	474.9	Input ON and sampling
			97.3	97.3	141.3	175.3	162.0	141.3	Input ON and not sampling
PID CONTROL WITH AUTOTUNING	PIDAT	4	446.3	446.3	712.5	765.3	700.0	---	First execution
			339.4	339.4	533.9	620.7	558.0	---	Input ON and sampling
			100.7	100.7	147.1	180.0	166.1	---	Input ON and not sampling
			189.2	189.2	281.6	233.7	225.1	---	Initial execution of autotuning
			535.2	535.2	709.8	575.3	558.2	---	Autotuning when sampling
LIMIT CONTROL	LMT	4 to 5	16.1	16.1	22.1	27.1	26.1	22.1	---
DEAD BAND CONTROL	BAND	4 to 5	17.0	17.0	22.5	27.4	26.6	22.5	---
DEAD ZONE CONTROL	ZONE	4 to 5	15.4	15.4	20.5	28.0	26.4	20.5	---
TIME-PROPORTIONAL OUTPUT	TPO	4	10.6	10.6	14.8	20.2	19.8	---	Input OFF
			54.5	54.5	82.0	92.7	85.1	---	Input ON and duty specified or output limit disabled
			61.0	61.0	91.9	102.5	95.3	---	Input ON, manipulated variable specified, and output limit disabled
SCALING	SCL	4	13.9	13.9	14.3	25.0	32.8	56.8	---
SCALING 2	SCL2	4	12.2	12.2	12.6	22.3	29.1	50.7	---
SCALING 3	SCL3	4	13.7	13.7	14.2	25.6	30.0	57.7	---
AVERAGE	AVG	4	36.3	36.3	52.6	62.9	59.1	53.1	Average of an operation
			291.0	291.0	419.9	545.3	492.7	419.9	Average of 64 operations

4-2-17 Subroutine Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□H-R	CJ1H CPU6□H	CJ1G CPU4□H	CJ1M CPU12/13/22/23	CJ1M CPU11/21	CJ1G CPU4□	
SUBROUTINE CALL	SBS	2	0.90	1.26	1.96	2.04	2.04	17.0	---
SUBROUTINE ENTRY	SBN	2	---	---	---	---	---	---	---
SUBROUTINE RETURN	RET	1	0.43	0.86	1.60	1.80	1.80	20.60	---
MACRO	MCRO	4	23.3	23.3	23.3	47.9	50.3	23.3	---
GLOBAL SUBROUTINE RETURN	GSBS	2	0.90	1.26	1.96	2.04	2.04	---	---
GLOBAL SUBROUTINE CALL	GSBN	2	---	---	---	---	---	---	---
GLOBAL SUBROUTINE ENTRY	GRET	1	0.43	0.86	1.60	1.80	1.80	---	---

4-2-18 Interrupt Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6☐ H-R	CJ1H CPU6☐ H	CJ1G CPU4☐ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4☐	
SET INTERRUPT MASK	MSKS	3	25.6	25.6	38.4	44.7	42.9	39.5	---
READ INTERRUPT MASK	MSKR	3	11.9	11.9	11.9	16.9	15.9	11.9	---
CLEAR INTERRUPT	CLI	3	27.4	27.4	41.3	42.7	44.5	41.3	---
DISABLE INTERRUPTS	DI	1	15.0	15.0	16.8	30.3	28.5	16.8	---
ENABLE INTERRUPTS	EI	1	19.5	19.5	21.8	37.7	34.4	21.8	---

4-2-19 High-speed Counter and Pulse Output Instructions (CJ1M-CPU21/22/23 Only)

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6☐ H-R	CJ1H CPU6☐ H	CJ1G CPU4☐ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4☐	
MODE CONTROL	INI	4	---	---	---	77.00	80.4	---	Starting high-speed counter comparison
			---	---	---	43.00	43.0	---	Stopping high-speed counter comparison
			---	---	---	43.40	48.8	---	Changing pulse output PV
			---	---	---	51.80	50.8	---	Changing high-speed counter PV
			---	---	---	31.83	28.5	---	Changing PV of counter in interrupt input mode
			---	---	---	45.33	49.8	---	Stopping pulse output
			---	---	---	36.73	30.5	---	Stopping PWM(891) output
HIGH-SPEED COUNTER PV READ	PRV	4	---	---	---	42.40	43.9	---	Reading pulse output PV
			---	---	---	53.40	65.9	---	Reading high-speed counter PV
			---	---	---	33.60	30.5	---	Reading PV of counter in interrupt input mode
			---	---	---	38.80	40.0	---	Reading pulse output status
			---	---	---	39.30	66.9	---	Reading high-speed counter status
			---	---	---	38.30	34.5	---	Reading PWM(891) status
			---	---	---	117.73	145.7	---	Reading high-speed counter range comparison results
			---	---	---	48.20	48.5	---	Reading frequency of high-speed counter 0
COUNTER FREQUENCY CONVERT	PRV2	4	---	---	---	23.03	22.39	---	---

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
COMPARI-SON TABLE LOAD	CTBL	4	---	---	---	238.0	235.0	---	Registering target value table and starting comparison for 1 target value
			---	---	---	14.42 ms	9.97 ms	---	Registering target value table and starting comparison for 48 target values
			---	---	---	289.0	276.0	---	Registering range table and starting comparison
			---	---	---	198.0	183.0	---	Only registering target value table for 1 target value
			---	---	---	14.40 ms	9.61 ms	---	Only registering target value table for 48 target values
			---	---	---	259.0	239.0	---	Only registering range table
SPEED OUTPUT	SPED	4	---	---	---	56.00	89.3	---	Continuous mode
			---	---	---	62.47	94.9	---	Independent mode
SET PULSES	PULS	4	---	---	---	26.20	32.9	---	---
PULSE OUTPUT	PLS2	5	---	---	---	100.80	107.5	---	---
ACCELERATION CONTROL	ACC	4	---	---	---	90.80	114.8	---	Continuous mode
			---	---	---	80.00	122.1	---	Independent mode
ORIGIN SEARCH	ORG	3	---	---	---	106.13	116.0	---	Origin search
			---	---	---	52.00	102.1	---	Origin return
PULSE WITH VARIABLE DUTY FACTOR	PWM	4	---	---	---	25.80	33.0	---	---

4-2-20 Step Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
STEP DEFINE	STEP	2	17.4	17.4	20.7	35.9	37.1	27.1	Step control bit ON
			11.8	11.8	13.7	13.8	18.3	24.4	Step control bit OFF
STEP START	SNXT	2	6.6	6.6	7.3	12.1	14.0	10.0	---

4-2-21 Basic I/O Unit Instructions

Instruction	Mne-monic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
I/O REFRESH	IORF	3	15.5	15.5	16.4	26.7	30.4	23.5	One input word
			17.20	17.20	18.40	29.7	35.0	25.6	One output word
SPECIAL I/O UNIT I/O REFRESH	FIORF	2	*1	---	---	---	---	---	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6☐ H-R	CJ1H CPU6☐ H	CJ1G CPU4☐ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4☐	
CPU BUS I/O REFRESH	DLNK	4	287.8	287.8	315.5	321.3	458.7	---	Allocated 1 word
7-SEGMENT DECODER	SDEC	4	6.5	6.5	6.9	8.1	15.7	14.1	---
DIGITAL SWITCH INPUT	DSW	6	50.7	50.7	73.5	77.7	77.6	---	4 digits, data input value: 0
			51.3	51.3	73.5	83.2	80.0	---	8 digits, data input value: 00
TEN KEY INPUT	TKY	4	9.7	9.7	13.2	18.7	18.6	---	Data input value: 00
			10.7	10.7	14.8	20.2	19.1	---	Data input value: FF
HEXADECIMAL KEY INPUT	HKY	5	50.3	50.3	70.9	77.3	78.1	---	Data input value: 00
			50.1	50.1	71.2	76.8	77.3	---	Data input value: FF
MATRIX INPUT	MTR	5	47.8	47.8	68.1	76.4	77.7	---	Data input value: 00
			48.0	48.0	68.0	77.7	76.9	---	Data input value: FF
7-SEGMENT DISPLAY OUTPUT	7SEG	5	58.1	58.1	83.3	89.6	89.9	---	4 digits
			63.3	63.3	90.3	98.3	99.2	---	8 digits
INTELLIGENT I/O READ	IORD	4	*1	*1	*1	*1	*1	*1	---
INTELLIGENT I/O WRITE	IOWR	4	*1	*1	*1	*1	*1	*1	---

*1 Execution of the IORD, IORW, and FIORF instructions depends on the Special I/O Units for which they are being executed.

4-2-22 Serial Communications Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6☐ H-R	CJ1H CPU6☐ H	CJ1G CPU4☐ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4☐	
PROTOCOL MACRO	PMCR	5	100.1	100.1	142.1	158.4	206.0	276.8	Direct specification
			134.2	134.2	189.6	210.0	256.7	305.9	Operand specification, sending 1 word, receiving 1 word
TRANSMIT	TXD	4	68.5	68.5	98.8	109.3	102.9	98.8	Sending 1 byte
			734.3	734.3	1.10 ms	1.23 ms	1.16 ms	1.10 ms	Sending 256 bytes
RECEIVE	RXD	4	89.6	89.6	131.1	144.0	132.1	131.1	Storing 1 byte
			724.2	724.2	1.11 ms	1.31 ms	1.22 ms	1.11 ms	Storing 256 bytes
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	4	131.5	131.5	202.4	213.4	208.6	---	Sending 1 byte

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	4	131	131	200.8	211.8	206.8	---	Storing 1 byte
CHANGE SERIAL PORT SETUP	STUP	3	341.2	341.2	400.0	504.7	524.7	440.4	Addressed to COM port on CPU Unit

4-2-23 Network Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
NETWORK SEND	SEND	4	84.4	84.4	123.9	141.6	195.0	123.9	---
NETWORK RECEIVE	RECV	4	85.4	85.4	124.7	142.3	196.7	124.7	---
DELIVER COMMAND	CMND	4	106.8	106.8	136.8	167.7	226.7	136.8	---
EXPLICIT MESSAGE SEND	EXPLT	4	127.6	127.6	190.0	217.0	238.0	---	---
EXPLICIT GET ATTRIBUTE	EGATR	4	123.9	123.9	185.0	210.0	232.7	---	---
EXPLICIT SET ATTRIBUTE	ESATR	3	110.0	110.0	164.4	188.3	210.3	---	---
EXPLICIT WORD READ	ECHRD	4	106.8	106.8	158.9	176.3	220.3	---	---
EXPLICIT WORD WRITE	ECHWR	4	106.0	106.0	158.3	175.7	205.3	---	---

4-2-24 File Memory Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
READ DATA FILE	FREAD	5	391.4	391.4	632.4	657.3	641.3	684.1	2-character directory + file name in binary
WRITE DATA FILE	FWRIT	5	387.8	387.8	627.0	650.7	637.3	684.7	2-character directory + file name in binary
WRITE TEXT FILE	TWRIT	5	390.1	390.1	619.1	555.3	489.0	---	---

4-2-25 Display Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
DISPLAY MESSAGE	MSG	3	10.1	10.1	14.2	16.8	17.3	14.3	Displaying message
			8.4	8.4	11.3	14.7	14.7	11.3	Deleting displayed message

4-2-26 Clock Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
CALENDAR ADD	CADD	4	34.0	38.3	201.9	217.0	194.0	209.5	---
CALENDAR SUBTRACT	CSUB	4	29.6	38.6	170.4	184.7	167.0	184.1	---
HOURS TO SECONDS	SEC	3	7.8	21.4	29.3	36.1	35.4	35.8	---
SECONDS TO HOURS	HMS	3	7.7	22.2	30.9	45.1	45.7	42.1	---
CLOCK ADJUSTMENT	DATE	2	216.0	216.0	251.5	118.7	128.3	120.0	---

4-2-27 Debugging Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
TRACE MEMORY SAMPLING	TRSM	1	80.4	80.4	120.0	207.0	218.3	120.0	Sampling 1 bit and 0 words
			848.1	848.1	1.06 ms	1.16 ms	1.10 ms	1.06 ms	Sampling 31 bits and 6 words

4-2-28 Failure Diagnosis Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
FAILURE ALARM	FAL	3	15.4	15.4	16.7	26.1	24.47	16.7	Recording errors
			179.8	179.8	244.8	294.0	264.0	244.8	Deleting errors (in order of priority)
			432.4	432.4	657.1	853.3	807.3	657.1	Deleting errors (all errors)
			161.5	161.5	219.4	265.7	233.0	219.4	Deleting errors (individually)
SEVERE FAILURE ALARM	FALS	3	---	---	---	---	---	---	---
FAILURE POINT DETECTION	FPD	4	140.9	140.9	202.3	220.7	250.0	202.3	Bit address output, time monitored
			163.4	163.4	217.6	250.3	264.3	217.6	Bit address output, first error detection
			185.2	185.2	268.9	220.7	321.7	268.9	Message characters output, time monitored
			207.5	207.5	283.6	320.7	336.0	283.6	Message characters output, first error detection

4-2-29 Other Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
SET CARRY	STC	1	0.048	0.06	0.06	0.15	0.15	0.12	---
CLEAR CARRY	CLC	1	0.048	0.06	0.06	0.15	0.15	0.12	---
SELECT EM BANK	EMBC	2	14.0	14.0	15.1	---	---	15.1	---
EXTEND MAXIMUM CYCLE TIME	WDT	2	15.0	15.0	19.7	23.6	22.0	19.7	---
SAVE CONDITION FLAGS	CCS	1	8.6	8.6	12.5	14.2	12.9	---	---
LOAD CONDITION FLAGS	CCL	1	9.8	9.8	13.9	16.3	15.7	---	---
CONVERT ADDRESS FROM CV	FRMCV	3	13.6	13.6	19.9	23.1	31.8	---	---
CONVERT ADDRESS TO CV	TOCV	3 to 4	11.9	11.9	17.2	22.5	31.4	---	---
DISABLE PERIPHERAL SERVICING	IOSP	1	13.9	13.9	19.8	21.5	21.5	---	---
ENABLE PERIPHERAL SERVICING	IORS	1	63.6	63.6	92.3	22.2	22.2	---	---

4-2-30 Block Programming Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
BLOCK PROGRAM BEGIN	BPRG	2	12.1	12.1	13.0	27.5	30.4	13.0	---
BLOCK PROGRAM END	BEND	1	9.6	9.6	12.3	23.2	27.1	13.1	---
BLOCK PROGRAM PAUSE	BPPS	2	10.6	10.6	12.3	16.0	21.7	14.9	---
BLOCK PROGRAM RESTART	BPRS	2	5.1	5.1	5.6	9.0	10.2	8.3	---
CONDITIONAL BLOCK EXIT	(Execution condition) EXIT	1	10.0	10.0	11.3	23.8	26.0	12.9	Block exited (input condition ON)
			4.0	4.0	4.9	7.2	8.4	7.3	Block not exited (input condition OFF)
CONDITIONAL BLOCK EXIT	EXIT (bit address)	2	9.8	9.8	13.5	28.4	30.6	16.3	Block exited (bit ON)
			4.7	4.7	7.2	11.4	13.1	10.7	Block not exited (bit OFF)
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT (bit address)	2	12.4	12.4	14.0	28.4	31.2	16.8	Block exited (bit OFF)
			7.1	7.1	7.6	11.8	13.5	11.2	Block not exited (bit ON)
Branching	IF (execution condition)	1	4.6	4.6	4.8	6.8	8.5	7.2	IF true (input condition ON)
			6.7	6.7	7.3	12.2	13.9	10.9	IF false (input condition OFF)
Branching	IF (bit address)	2	6.8	6.8	7.2	11.0	12.7	10.4	IF true (bit ON)
			9.0	9.0	9.6	16.5	18.5	14.2	IF false (bit OFF)
Branching (NOT)	IF NOT (bit address)	2	7.1	7.1	7.6	11.5	13.1	10.9	IF true (bit OFF)
			9.2	9.2	10.1	16.8	18.9	14.7	IF false (bit ON)

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
Branching	ELSE	1	6.2	6.2	6.7	11.4	12.6	9.9	IF true
			6.8	6.8	7.7	13.4	15.0	11.2	IF false
Branching	IEND	1	6.9	6.9	7.7	13.5	15.4	11.0	IF true
			4.4	4.4	4.6	6.93	8.1	7.0	IF false
ONE CYCLE AND WAIT	WAIT (execution condition)	1	12.6	12.6	13.7	28.6	34.0	16.7	Do not wait (input condition ON)
			3.9	3.9	4.1	5.6	6.9	6.3	Wait (input condition OFF)
ONE CYCLE AND WAIT	WAIT (bit address)	2	12.0	12.0	13.4	27.2	30.0	16.5	Do not wait (bit ON)
			6.1	6.1	6.5	10.0	11.4	9.6	Wait (bit OFF)
ONE CYCLE AND WAIT (NOT)	WAIT NOT (bit address)	2	12.2	12.2	13.8	27.8	30.6	17.0	Do not wait (bit OFF)
			6.4	6.4	6.9	10.5	11.8	10.1	Wait (bit ON)
HUNDRED-MS TIMER WAIT	TIMW	3	22.3	22.3	25.2	47.4	52.0	33.1	Default setting
			24.9	24.9	27.8	46.2	53.4	35.7	Normal execution
	TIMWX	3	22.3	22.3	25.2	47.4	52.0	33.1	Default setting
			24.9	24.9	27.8	46.2	53.4	35.7	Normal execution
TEN-MS TIMER WAIT	TMHW	3	25.8	25.8	27.9	47.9	53.7	34.1	Default setting
			20.6	20.6	22.7	40.9	46.2	28.9	Normal execution
	TMHWX	3	25.8	25.8	27.9	47.9	53.7	34.1	Default setting
			20.6	20.6	22.7	40.9	46.2	28.9	Normal execution
COUNTER WAIT	CNTW	4	17.9	17.9	22.6	41.0	43.5	27.4	Default setting
			19.1	19.1	23.9	42.9	45.7	28.7	Normal execution
	CNTWX	4	17.9	17.9	22.6	41.0	43.5	27.4	Default setting
			19.1	19.1	23.9	42.9	45.7	28.7	Normal execution
Loop Control	LOOP	1	7.9	7.9	9.1	15.6	17.6	12.3	---
Loop Control	LEND (execution condition)	1	7.7	7.7	8.4	13.5	15.5	10.9	Do not loop (input condition ON)
			6.8	6.8	8.0	17.5	19.8	9.8	Loop (input condition OFF)
Loop Control	LEND (bit address)	2	9.9	9.9	10.7	17.5	19.9	14.4	Do not loop (bit ON)
			8.9	8.9	10.3	21.6	24.5	13.0	Loop (bit OFF)
Loop Control (NOT)	LEND NOT (bit address)	2	10.2	10.2	11.2	21.9	24.9	14.8	Do not loop (bit OFF)
			9.3	9.3	10.8	17.8	20.4	13.5	Loop (bit ON)

4-2-31 Text String Processing Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
MOV STRING	MOV$	3	45.6	45.6	66.0	79.3	72.7	84.3	Transferring 1 character*1
CONCATENATE STRING	+$	4	86.5	86.5	126.0	152.0	137.0	167.8	1 character + 1 character*1
GET STRING LEFT	LEFT$	4	53.0	53.0	77.4	93.6	84.8	94.3	Retrieving 1 character from 2 characters*1
GET STRING RIGHT	RGHT$	4	52.2	52.2	76.3	92.1	83.3	94.2	Retrieving 1 character from 2 characters*1

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
GET STRING MIDDLE	MID$	5	56.5	56.5	84.6	93.7	84.0	230.2	Retrieving 1 character from 3 characters[*1]
FIND IN STRING	FIND$	4	51.4	51.4	77.5	89.1	96.7	94.1	Searching for 1 character from 2 characters[*1]
STRING LENGTH	LEN$	3	19.8	19.8	28.9	33.8	30.1	33.4	Detecting 1 character[*1]
REPLACE IN STRING	RPLC$	6	175.1	175.1	258.7	300.7	267.7	479.5	Replacing the first of 2 characters with 1 character[*1]
DELETE STRING	DEL$	5	63.4	63.4	94.2	111.3	99.3	244.6	Deleting the leading character of 2 characters[*1]
EXCHANGE STRING	XCHG$	3	60.6	60.6	87.2	105.2	95.3	99.0	Exchanging 1 character with 1 character[*1]
CLEAR STRING	CLR$	2	23.8	23.8	36.0	42.0	36.8	37.8	Clearing 1 character[*1]
INSERT INTO STRING	INS$	5	136.5	136.5	200.6	204.0	208.0	428.9	Inserting 1 character after the first of 2 characters[*1]
String Comparison Instructions	=$	4	48.5	48.5	69.8	79.9	68.5	86.2	Comparing 1 character with 1 character
	<>$								
	<$								
	>$								
	=$								

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-2-32 Task Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/13/ 22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
TASK ON	TKON	2	6.4	6.4	10.2	11.2	9.55	3.2	Cyclic task specified
			20.6	20.6	30.1	34.6	34.9	---	Extra task specified
TASK OFF	TKOF	2	123.7	123.7	195.8	214.5	183.4	61.3	Cyclic task specified
			23.4	23.4	34.1	39.3	39.6	---	Extra task specified

4-2-33 Model Conversion Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
BLOCK TRANSFER	XFERC	4	6.4	6.4	6.5	33.1	31.1	---	Transferring 1 word
			481.6	481.6	791.6	3056.1	2821.1	---	Transferring 1,000 words
SINGLE WORD DISTRIBUTE	DISTC	4	3.4	3.4	3.5	19	18.1	---	Data distribute
			5.9	5.9	7.3	39.5	38.5	---	Stack operation
DATA COLLECT	COLLC	4	3.5	3.5	3.85	24.9	29.7	---	Data distribute
			8	8	9.1	22.1	25.3	---	Stack operation
			8.3	8.3	9.6	25.5	31	---	Stack operation 1 word FIFO Read
			2052.3	2052.3	2097.5	8310.1	7821.1	---	Stack operation 1,000 word FIFO Read
MOVE BIT	MOVBC	4	4.5	4.5	4.88	28.1	22.1	---	---
BIT COUNTER	BCNTC	4	4.9	4.9	5	30.6	28.8	---	Counting 1 word
			1252.4	1252.4	1284.4	5814.1	5223.8	---	Counting 1,000 words

4-2-34 Special Function Block Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)						Conditions
			CJ1H CPU6□ H-R	CJ1H CPU6□ H	CJ1G CPU4□ H	CJ1M CPU12/ 13/22/23	CJ1M CPU11/ 21	CJ1G CPU4□	
GET VARIABLE ID	GETID	4	14	14	22.2	23.4	21.3	---	---

4-2-35 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 using CJ1-H CPU Units and CJ1M CPU Units with unit version 3.0 or later.

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			CPU Unit model			
			CJ1H-CPU6□H-R	**CJ1H-CPU6□H**	**CJ1G-CPU4□H**	**CJ1M-CPU□□**
A	Startup time	Startup time not including I/O parameter transfer	3.3 µs	6.8 µs	8.8 µs	15.0 µs
B	I/O parameter transfer processing time The data type is indicated in parentheses.	1-bit (BOOL) input symbol or output symbol	0.24 µs	0.4 µs	0.7 µs	1.0 µs
		1-word (INT, UINT, WORD) input symbol or output symbol	0.19 µs	0.3 µs	0.6 µs	0.8 µs
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	0.19 µs	0.5 µs	0.8 µs	1.1 µs
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	0.38 µs	1.0 µs	1.6 µs	2.2 µs
		I/O symbols	0.114 µs	0.4 µs	0.5 µs	1.2 µs
C	Function block definition instruction execution time	Total instruction processing time (same as standard user program)				

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.

Note Number of Function Block Program Steps (CJ1-H CPU Units and CJ1M CPU Units with unit version 3.0 or later)

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 using CJ1-H CPU Units and CJ1M CPU Units with unit version 3.0 or later.

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			**CJ1-H CPU Units, CJ1M CPU Units with unit version 3.0 or later**
m	Call part		57 steps
n	I/O parameter transfer part The data type is shown in parentheses.	1-bit (BOOL) input symbol or output symbol	6 steps
		1-word (INT, UINT, WORD) input symbol or output symbol	6 steps
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	6 steps
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	18 steps
		I/O symbols	6 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

4-3 CS-series Instruction Execution Times and Number of Steps

The following table lists the execution times for all instructions that are available for CS-series CPU Units (except for CS1D CPU Units[*1]).

[*1] Refer to the *SYSMAC CS1D Duplex System Operation Manual* (Cat. No. W405) for the execution times for CS1D CPU Units.

The total execution time of instructions within one whole user program is the process time for program execution when calculating the cycle time (See note.).

Note User programs are allocated tasks that can be executed within cyclic tasks and interrupt tasks that satisfy interrupt conditions.

Execution times for most instructions differ depending on the CPU Unit used (CS1H-CPU6□H, CS1H-CPU6□, CS1G-CPU4□H, CS1G-CPU4□) and the conditions when the instruction is executed.

The execution time can also vary when the execution condition is OFF.

The following table also lists the length of each instruction in the *Length (steps)* column. The number of steps required in the user program area for each of the CS-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 CS-series PLCs is measured in steps, whereas program capacity for previous OMRON PLCs, such as the C-series and CV-series PLCs, was measured in words. Basically speaking, 1 step is equivalent to 1 word. The amount of memory required for each instruction, however, is different for some of the CS-series instructions, and inaccuracies will occur if the capacity of a user program for another PLC is converted for a CS-series PLC based on the assumption that 1 word is 1 step. Refer to the information at the end of *4-3 CS-series Instruction Execution Times and Number of Steps* for guidelines on converting program capacities from previous OMRON PLCs.

   Most instructions are supported in differentiated form (indicated with ↑, ↓, @, and %). Specifying differentiation will increase the execution times by the following amounts.

Symbol	CS1-H CPU Units		CS1 CPU Units	
	CPU6□H (μs)	CPU4□H (μs)	CPU6□ (μs)	CPU4□ (μs)
↑ or ↓	+0.24	+0.32	+0.41	+0.45
@ or %	+0.24	+0.32	+0.29	+0.33

2. Use the following times as guidelines when instructions are not executed.

CS1-H CPU Units		CS1 CPU Units	
CPU6□H (μs)	CPU4□H (μs)	CPU6□ (μs)	CPU4□ (μs)
0.018 to 0.108	0.02 to 0.12	0.08 to 0.48	0.12 to 0.72

4-3-1 Sequence Input Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
LOAD	LD	1	0.02	0.04	0.04	0.08	---
	!LD	2	21.16	21.20	21.20	21.24	CS Series
			45.12	45.14	45.14	45.18	C200H
LOAD NOT	LD NOT	1	0.02	0.04	0.04	0.08	---
	!LD NOT	2	21.16	21.20	21.20	21.24	CS Series
			45.12	45.14	45.14	45.18	C200H
AND	AND	1	0.02	0.04	0.04	0.08	---
	!AND	2	21.16	21.20	21.20	21.24	CS Series
			45.12	45.14	45.14	45.18	C200H
AND NOT	AND NOT	1	0.02	0.04	0.04	0.08	---
	!AND NOT	2	21.16	21.20	21.20	21.24	CS Series
			45.12	45.14	45.14	45.18	C200H
OR	OR	1	0.02	0.04	0.04	0.08	---
	!OR	2	21.16	21.20	21.20	21.24	CS Series
			45.12	45.14	45.14	45.18	C200H
OR NOT	OR NOT	1	0.02	0.04	0.04	0.08	---
	!OR NOT	2	21.16	21.20	21.20	21.24	CS Series
			45.12	45.14	45.14	45.18	C200H
AND LOAD	AND LD	1	0.02	0.04	0.04	0.08	---
OR LOAD	OR LD	1	0.02	0.04	0.04	0.08	---
NOT	NOT	1	0.02	0.04	0.04	0.08	---
CONDITION ON	UP	3	0.3	0.42	0.46	0.54	---
CONDITION OFF	DOWN	4	0.3	0.42	0.46	0.54	---
LOAD BIT TEST	LD TST	4	0.14	0.24	0.25	0.37	---
LOAD BIT TEST NOT	LD TSTN	4	0.14	0.24	0.25	0.37	---
AND BIT TEST NOT	AND TSTN	4	0.14	0.24	0.25	0.37	---
OR BIT TEST	OR TST	4	0.14	0.24	0.25	0.37	---
OR BIT TEST NOT	OR TSTN	4	0.14	0.24	0.25	0.37	---

4-3-2 Sequence Output Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
OUTPUT	OUT	1	0.02	0.04	0.17	0.21	---
	!OUT	2	21.39	21.41	21.54	21.58	CS Series
			49.32	49.34	49.47	49.51	C200H
OUTPUT NOT	OUT NOT	1	0.02	0.04	0.17	0.21	---
	!OUT NOT	2	21.39	21.41	21.54	21.58	CS Series
			49.32	49.34	49.47	49.51	C200H
KEEP	KEEP	1	0.06	0.08	0.25	0.29	---
DIFFERENTIATE UP	DIFU	2	0.24	0.40	0.46	0.54	---
DIFFERENTIATE DOWN	DIFD	2	0.24	0.40	0.46	0.54	---
SET	SET	1	0.02	0.06	0.17	0.21	---
	!SET	2	21.39	21.41	21.54	21.58	CS Series
			49.32	49.34	49.47	49.51	C200H

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
RESET	RSET	1	0.02	0.06	0.17	0.21	Word specified
	!RSET	2	21.39	21.41	21.54	21.58	CS Series
			49.32	49.34	49.47	49.51	C200H
MULTIPLE BIT SET	SETA	4	5.8	6.1	7.8	7.8	With 1-bit set
			25.7	27.2	38.8	38.8	With 1,000-bit set
MULTIPLE BIT RESET	RSTA	4	5.7	6.1	7.8	7.8	With 1-bit reset
			25.8	27.1	38.8	38.8	With 1,000-bit reset
SINGLE BIT SET	SETB	2	0.24	0.34	---	---	---
	!SETB	3	21.48	21.88	---	---	---
SINGLE BIT RESET	RSTB	2	0.24	0.34	---	---	---
	!RSTB	3	21.68	21.88	---	---	---
SINGLE BIT OUTPUT	OUTB	2	0.22	0.32	---	---	---
	!OUTB	3	21.64	21.84	---	---	---

4-3-3 Sequence Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
END	END	1	5.5	6.0	4.0	4.0	---
NO OPERATION	NOP	1	0.02	0.04	0.08	0.12	---
INTERLOCK	IL	1	0.06	0.06	0.12	0.12	---
INTERLOCK CLEAR	ILC	1	0.06	0.06	0.12	0.12	---
MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	3	6.1	6.5	---	---	Interlock condition not met (input condition ON)
			7.5	7.9	---	---	Interlock condition met (input condition OFF)
			8.9	9.7	---	---	Interlock condition met again during interlock (input condition OFF)
MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	3	6.1	6.5	---	---	Interlock condition not met (input condition ON)
			7.5	7.9	---	---	Interlock condition met (input condition OFF)
			8.9	9.7	---	---	Interlock condition met again during interlock (input condition OFF)
MULTI-INTERLOCK CLEAR	MILC	2	5.0	5.6	---	---	Not during interlock
			5.7	6.2	---	---	During interlock
JUMP	JMP	2	0.38	0.48	8.1	8.1	---
JUMP END	JME	2	---	---	---	---	---
CONDITIONAL JUMP	CJP	2	0.38	0.48	7.4	7.4	Jump condition met (input condition ON)
CONDITIONAL JUMP NOT	CJPN	2	0.38	0.48	8.5	8.5	Jump condition met (input condition OFF)
MULTIPLE JUMP	JMP0	1	0.06	0.06	0.12	0.12	---
MULTIPLE JUMP END	JME0	1	0.06	0.06	0.12	0.12	---
FOR LOOP	FOR	2	0.12	0.21	0.12	0.21	Designating a constant
BREAK LOOP	BREAK	1	0.12	0.12	0.12	0.12	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
NEXT LOOP	NEXT	1	0.17	0.17	0.17	0.17	When loop is continued
			0.12	0.12	0.12	0.12	When loop is ended

4-3-4 Timer and Counter Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
HUNDRED-MS TIMER	TIM	3	0.56	0.88	0.37	0.42	---
	TIMX	3	0.56	0.88	0.37	0.42	---
TEN-MS TIMER	TIMH	3	0.88	1.14	0.37	0.42	---
	TIMHX	3	0.88	1.14	0.37	0.42	---
ONE-MS TIMER	TMHH	3	0.86	1.12	0.37	0.42	---
	TMHHX	3	0.86	1.12	0.37	0.42	---
ACCUMULATIVE TIMER	TTIM	3	16.1	17.0	21.4	21.4	---
			10.9	11.4	14.8	14.8	When resetting
			8.5	8.7	10.7	10.7	When interlocking
	TTIMX	3	16.1	17.0	21.4	21.4	---
			10.9	11.4	14.8	14.8	When resetting
			8.5	8.7	10.7	10.7	When interlocking
LONG TIMER	TIML	4 to 5	7.6	10.0	12.8	12.8	---
			6.2	6.5	7.8	7.8	When interlocking
	TIMLX	4 to 5	7.6	10.0	12.8	12.8	---
			6.2	6.5	7.8	7.8	When interlocking
MULTI-OUTPUT TIMER	MTIM	4	20.9	23.3	26.0	26.0	---
			5.6	5.8	7.8	7.8	When resetting
	MTIMX	4	20.9	23.3	26.0	26.0	---
			5.6	5.8	7.8	7.8	When resetting
COUNTER	CNT	3	0.56	0.88	0.37	0.42	---
	CNTX	3	0.56	0.88	0.37	0.42	---
REVERSIBLE COUNTER	CNTR	3	16.9	19.0	20.9	20.9	---
	CNTRX	3	16.9	19.0	20.9	20.9	---
RESET TIMER/ COUNTER	CNR	3	9.9	10.6	13.9	13.9	When resetting 1 word
			4160	4160	5420	5420	When resetting 1,000 words
	CNRX	3	9.9	10.6	13.9	13.9	When resetting 1 word
			4160	4160	5420	5420	When resetting 1,000 words

4-3-5 Comparison Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
Input Comparison Instructions (unsigned)	=	4	0.10	0.16	0.21	0.37	---
	<>						
	<						
	<=						
	>						
	>=						
Input Comparison Instructions (double, unsigned)	=L	4 to 7	0.10	0.16	0.29	0.54	---
	<>L						
	<L						
	<=L						
	>L						
	>=L						
Input Comparison Instructions (signed)	=S	4	0.10	0.16	6.50	6.50	---
	<>S						
	<S						
	<=S						
	>S						
	>=S						
Input Comparison Instructions (double, signed)	=SL	4 to 7	0.10	0.16	6.50	6.50	---
	<>SL						
	<SL						
	<=SL						
	>SL						
	>=SL						
Time Comparison Instructions	=DT	4	25.1	36.4	---	---	---
	<>DT	4	25.2	36.4	---	---	
	<DT	4	25.2	36.4	---	---	
	<=DT	4	25.2	36.4	---	---	
	>DT	4	25.1	36.4	---	---	
	>=DT	4	25.2	36.4	---	---	
COMPARE	CMP	3	0.04	0.04	0.17	0.29	---
	!CMP	7	42.14	42.14	42.57	42.69	CS Series
			90.44	90.44	90.67	90.79	CS Series
DOUBLE COMPARE	CMPL	3 to 5	0.08	0.08	0.25	0.46	---
SIGNED BINARY COMPARE	CPS	3	0.08	0.08	6.50	6.50	---
	!CPS	7	35.98	35.98	48.9	48.9	CS Series
			84.18	84.18	97.0	97.0	C200H
DOUBLE SIGNED BINARY COMPARE	CPSL	3 to 5	0.08	0.08	6.50	6.50	---
TABLE COMPARE	TCMP	4	14.0	15.2	21.9	21.9	---
MULTIPLE COMPARE	MCMP	4	20.5	22.8	31.2	31.2	---
UNSIGNED BLOCK COMPARE	BCMP	4	21.5	23.7	32.6	32.6	---
EXPANDED BLOCK COMPARE	BCMP2	4	8.4	9.3	---	---	Number of data words: 1
			313.0	345.3	---	---	Number of data words: 255

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
AREA RANGE COMPARE	ZCP	3	5.3	5.4	---	---	---
DOUBLE AREA RANGE COMPARE	ZCPL	3 to 5	5.5	6.7	---	---	---

4-3-6 Data Movement Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
MOVE	MOV	3	0.18	0.20	0.25	0.29	---
	!MOV	7	21.56	21.60	42.61	42.65	CS Series
			90.70	90.72	90.77	90.83	C200H
DOUBLE MOVE	MOVL	3 to 4	0.32	0.34	0.42	0.50	---
MOVE NOT	MVN	3	0.18	0.20	0.25	0.29	---
DOUBLE MOVE NOT	MVNL	3 to 4	0.32	0.34	0.42	0.50	---
MOVE BIT	MOVB	4	0.24	0.34	7.5	7.5	---
MOVE DIGIT	MOVD	4	0.24	0.34	7.3	7.3	---
MULTIPLE BIT TRANSFER	XFRB	4	10.1	10.8	13.6	13.6	Transferring 1 bit
			186.4	189.8	269.2	269.2	Transferring 255 bits
BLOCK TRANSFER	XFER	4	0.36	0.44	11.2	11.2	Transferring 1 word
			300.1	380.1	633.5	633.5	Transferring 1,000 words
BLOCK SET	BSET	4	0.26	0.28	8.5	8.5	Setting 1 word
			200.1	220.1	278.3	278.3	Setting 1,000 words
DATA EXCHANGE	XCHG	3	0.40	0.56	0.5	0.7	---
DOUBLE DATA EXCHANGE	XCGL	3 to 4	0.76	1.04	0.9	1.3	---
SINGLE WORD DISTRIBUTE	DIST	4	5.1	5.4	7.0	7.0	---
DATA COLLECT	COLL	4	5.1	5.3	7.1	7.1	---
MOVE TO REGISTER	MOVR	3	0.08	0.08	0.42	0.50	---
MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	3	0.42	0.50	0.42	0.50	---

4-3-7 Data Shift Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SHIFT REGISTER	SFT	3	7.4	10.4	10.4	10.4	Shifting 1 word
			433.2	488.0	763.1	763.1	Shifting 1,000 words
REVERSIBLE SHIFT REGISTER	SFTR	4	6.9	7.2	9.6	9.6	Shifting 1 word
			615.3	680.2	859.6	859.6	Shifting 1,000 words
ASYNCHRONOUS SHIFT REGISTER	ASFT	4	6.2	6.4	7.7	7.7	Shifting 1 word
			1220	1220	2010	2010	Shifting 1,000 words[*1]
WORD SHIFT	WSFT	4	4.5	4.7	7.8	7.8	Shifting 1 word
			171.5	171.7	781.7	781.7	Shifting 1,000 words
ARITHMETIC SHIFT LEFT	ASL	2	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT LEFT	ASLL	2	0.40	0.56	0.50	0.67	---

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
ARITHMETIC SHIFT RIGHT	ASR	2	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT RIGHT	ASRL	2	0.40	0.56	0.50	0.67	---
ROTATE LEFT	ROL	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE LEFT	ROLL	2	0.40	0.56	0.50	0.67	---
ROTATE LEFT WITHOUT CARRY	RLNC	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	2	0.40	0.56	0.50	0.67	---
ROTATE RIGHT	ROR	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE RIGHT	RORL	2	0.40	0.56	0.50	0.67	---
ROTATE RIGHT WITHOUT CARRY	RRNC	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	2	0.40	0.56	0.50	0.67	---
ONE DIGIT SHIFT LEFT	SLD	3	5.9	6.1	8.2	8.2	Shifting 1 word
			561.1	626.3	760.7	760.7	Shifting 1,000 words
ONE DIGIT SHIFT RIGHT	SRD	3	6.9	7.1	8.7	8.7	Shifting 1 word
			760.5	895.5	1070	1070	Shifting 1,000 words
SHIFT N-BIT DATA LEFT	NSFL	4	7.5	8.3	10.5	10.5	Shifting 1 bit
			40.3	45.4	55.5	55.5	Shifting 1,000 bits
SHIFT N-BIT DATA RIGHT	NSFR	4	7.5	8.3	10.5	10.5	Shifting 1 bit
			50.5	55.3	69.3	69.3	Shifting 1,000 bits
SHIFT N-BITS LEFT	NASL	3	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT N-BITS LEFT	NSLL	3	0.40	0.56	0.50	0.67	---
SHIFT N-BITS RIGHT	NASR	3	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	3	0.40	0.56	0.50	0.67	---

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-3-8 Increment/Decrement Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
INCREMENT BINARY	++	2	0.22	0.32	0.29	0.37	---
DOUBLE INCREMENT BINARY	++L	2	0.40	0.56	0.50	0.67	---
DECREMENT BINARY	– –	2	0.22	0.32	0.29	0.37	---
DOUBLE DECREMENT BINARY	– –L	2	0.40	0.56	0.50	0.67	---
INCREMENT BCD	++B	2	6.4	4.5	7.4	7.4	---
DOUBLE INCREMENT BCD	++BL	2	5.6	4.9	6.1	6.1	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
DECREMENT BCD	– –B	2	6.3	4.6	7.2	7.2	---
DOUBLE DECREMENT BCD	– –BL	2	5.3	4.7	7.1	7.1	---

4-3-9 Symbol Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SIGNED BINARY ADD WITHOUT CARRY	+	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	4 to 6	0.32	0.34	0.42	0.54	---
SIGNED BINARY ADD WITH CARRY	+C	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	4 to 6	0.32	0.34	0.42	0.54	---
BCD ADD WITHOUT CARRY	+B	4	8.2	8.4	14.0	14.0	---
DOUBLE BCD ADD WITHOUT CARRY	+BL	4 to 6	13.3	14.5	19.0	19.0	---
BCD ADD WITH CARRY	+BC	4	8.9	9.1	14.5	14.5	---
DOUBLE BCD ADD WITH CARRY	+BCL	4 to 6	13.8	15.0	19.6	19.6	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	4 to 6	0.32	0.34	0.42	0.54	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	4 to 6	0.32	0.34	0.42	0.54	---
BCD SUBTRACT WITHOUT CARRY	–B	4	8.0	8.2	13.1	13.1	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	4 to 6	12.8	14.0	18.2	18.2	---
BCD SUBTRACT WITH CARRY	–BC	4	8.5	8.6	13.8	13.8	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	4 to 6	13.4	14.7	18.8	18.8	---
SIGNED BINARY MULTIPLY	*	4	0.38	0.40	0.50	0.58	---
DOUBLE SIGNED BINARY MULTIPLY	*L	4 to 6	7.23	8.45	11.19	11.19	---
UNSIGNED BINARY MULTIPLY	*U	4	0.38	0.40	0.50	0.58	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	4 to 6	7.1	8.3	10.63	10.63	---
BCD MULTIPLY	*B	4	9.0	9.2	12.8	12.8	---
DOUBLE BCD MULTIPLY	*BL	4 to 6	23.0	24.2	35.2	35.2	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SIGNED BINARY DIVIDE	/	4	0.40	0.42	0.75	0.83	---
DOUBLE SIGNED BINARY DIVIDE	/L	4 to 6	7.2	8.4	9.8	9.8	---
UNSIGNED BINARY DIVIDE	/U	4	0.40	0.42	0.75	0.83	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	4 to 6	6.9	8.1	9.1	9.1	---
BCD DIVIDE	/B	4	8.6	8.8	15.9	15.9	---
DOUBLE BCD DIVIDE	/BL	4 to 6	17.7	18.9	26.2	26.2	---

4-3-10 Conversion Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
BCD TO BINARY	BIN	3	0.22	0.24	0.25	0.29	---
DOUBLE BCD TO DOUBLE BINARY	BINL	3 to 4	6.5	6.8	9.1	9.1	---
BINARY TO BCD	BCD	3	0.24	0.26	8.3	8.3	---
DOUBLE BINARY TO DOUBLE BCD	BCDL	3 to 4	6.7	7.0	9.2	9.2	---
2'S COMPLEMENT	NEG	3	0.18	0.20	0.25	0.29	---
DOUBLE 2'S COMPLEMENT	NEGL	3 to 4	0.32	0.34	0.42	0.50	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN	3	0.32	0.34	0.42	0.50	---
DATA DECODER	MLPX	4	0.32	0.42	8.8	8.8	Decoding 1 digit (4 to 16)
			0.98	1.20	12.8	12.8	Decoding 4 digits (4 to 16)
			3.30	4.00	20.3	20.3	Decoding 1 digit 8 to 256
			6.50	7.90	33.4	33.4	Decoding 4 digits (8 to 256)
DATA ENCODER	DMPX	4	7.5	7.9	10.4	10.4	Encoding 1 digit (16 to 4)
			49.6	50.2	59.1	59.1	Encoding 4 digits (16 to 4)
			18.2	18.6	23.6	23.6	Encoding 1 digit (256 to 8)
			55.1	57.4	92.5	92.5	Encoding 2 digits (256 to 8)
ASCII CONVERT	ASC	4	6.8	7.1	9.7	9.7	Converting 1 digit into ASCII
			11.2	11.7	15.1	15.1	Converting 4 digits into ASCII
ASCII TO HEX	HEX	4	7.1	7.4	10.1	10.1	Converting 1 digit
COLUMN TO LINE	LINE	4	19.0	23.1	29.1	29.1	---
LINE TO COLUMN	COLM	4	23.2	27.5	37.3	37.3	---

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SIGNED BCD TO BINARY	BINS	4	8.0	8.3	12.1	12.1	Data format setting No. 0
			8.0	8.3	12.1	12.1	Data format setting No. 1
			8.3	8.6	12.7	12.7	Data format setting No. 2
			8.5	8.8	13.0	13.0	Data format setting No. 3
DOUBLE SIGNED BCD TO BINARY	BISL	4 to 5	9.2	9.6	13.6	13.6	Data format setting No. 0
			9.2	9.6	13.7	13.7	Data format setting No. 1
			9.5	9.9	14.2	14.2	Data format setting No. 2
			9.6	10.0	14.4	14.4	Data format setting No. 3
SIGNED BINARY TO BCD	BCDS	4	6.6	6.9	10.6	10.6	Data format setting No. 0
			6.7	7.0	10.8	10.8	Data format setting No. 1
			6.8	7.1	10.9	10.9	Data format setting No. 2
			7.2	7.5	11.5	11.5	Data format setting No. 3
DOUBLE SIGNED BINARY TO BCD	BDSL	4 to 5	8.1	8.4	11.6	11.6	Data format setting No. 0
			8.2	8.6	11.8	11.8	Data format setting No. 1
			8.3	8.7	12.0	12.0	Data format setting No. 2
			8.8	9.2	12.5	12.5	Data format setting No. 3
GRAY CODE CONVERSION	GRY	4	46.9	72.1	---	---	8-bit binary
			49.6	75.2	---	---	8-bit BCD
			57.7	87.7	---	---	8-bit angle
			61.8	96.7	---	---	15-bit binary
			64.5	99.6	---	---	15-bit BCD
			72.8	112.4	---	---	15-bit angle
			52.3	87.2	---	---	360° binary
			55.1	90.4	---	---	360° BCD
			64.8	98.5	---	---	360° angle
FOUR-DIGIT NUMBER TO ASCII	STR4	3	13.79	20.24	---	---	
EIGHT-DIGIT NUMBER TO ASCII	STR8	3 to 4	18.82	27.44	---	---	
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	3	30.54	44.41	---	---	
ASCII TO FOUR-DIGIT NUMBER	NUM4	3 to 4	18.46	27.27	---	---	
ASCII TO EIGHT-DIGIT NUMBER	NUM8	3	27.27	40.29	---	---	
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	3	52.31	78.25	---	---	

4-3-11 Logic Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
LOGICAL AND	ANDW	4	0.18	0.20	0.25	0.37	---
DOUBLE LOGICAL AND	ANDL	4 to 6	0.32	0.34	0.42	0.54	---
LOGICAL OR	ORW	4	0.22	0.32	0.25	0.37	---
DOUBLE LOGICAL OR	ORWL	4 to 6	0.32	0.34	0.42	0.54	---
EXCLUSIVE OR	XORW	4	0.22	0.32	0.25	0.37	---
DOUBLE EXCLUSIVE OR	XORL	4 to 6	0.32	0.34	0.42	0.54	---
EXCLUSIVE NOR	XNRW	4	0.22	0.32	0.25	0.37	---
DOUBLE EXCLUSIVE NOR	XNRL	4 to 6	0.32	0.34	0.42	0.54	---
COMPLEMENT	COM	2	0.22	0.32	0.29	0.37	---
DOUBLE COMPLEMENT	COML	2	0.40	0.56	0.50	0.67	---

4-3-12 Special Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
BINARY ROOT	ROTB	3	49.6	50.0	530.7	530.7	---
BCD SQUARE ROOT	ROOT	3	13.7	13.9	514.5	514.5	---
ARITHMETIC PROCESS	APR	4	6.7	6.9	32.3	32.3	Designating SIN and COS
			17.2	18.4	78.3	78.3	Designating line-segment approximation (with two coordinates in the line-segment data table)
FLOATING POINT DIVIDE	FDIV	4	116.6	176.6	176.6	176.6	---
BIT COUNTER	BCNT	4	0.3	0.38	22.1	22.1	Counting 1 word

4-3-13 Floating-point Math Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
FLOATING TO 16-BIT	FIX	3 to 4	10.6	10.8	14.5	14.5	---
FLOATING TO 32-BIT	FIXL	3 to 4	10.8	11.0	14.6	14.6	---
16-BIT TO FLOATING	FLT	3 to 4	8.3	8.5	11.1	11.1	---
32-BIT TO FLOATING	FLTL	3 to 4	8.3	8.5	10.8	10.8	---
FLOATING-POINT ADD	+F	4 to 6	8.0	9.2	10.2	10.2	---
FLOATING-POINT SUBTRACT	–F	4 to 6	8.0	9.2	10.3	10.3	---
FLOATING-POINT DIVIDE	/F	4 to 6	8.7	9.9	12.0	12.0	---
FLOATING-POINT MULTIPLY	*F	4 to 6	8.0	9.2	10.5	10.5	---
DEGREES TO RADIANS	RAD	3 to 4	10.1	10.2	14.9	14.9	---
RADIANS TO DEGREES	DEG	3 to 4	9.9	10.1	14.8	14.8	---

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SINE	SIN	3 to 4	42.0	42.2	61.1	61.1	---
COSINE	COS	3 to 4	31.5	31.8	44.1	44.1	---
TANGENT	TAN	3 to 4	16.3	16.6	22.6	22.6	---
ARC SINE	ASIN	3 to 4	17.6	17.9	24.1	24.1	---
ARC COSINE	ACOS	3 to 4	20.4	20.7	28.0	28.0	---
ARC TANGENT	ATAN	3 to 4	16.1	16.4	16.4	16.4	---
SQUARE ROOT	SQRT	3 to 4	19.0	19.3	28.1	28.1	---
EXPONENT	EXP	3 to 4	65.9	66.2	96.7	96.7	---
LOGARITHM	LOG	3 to 4	12.8	13.1	17.4	17.4	---
EXPONENTIAL POWER	PWR	4 to 6	125.4	126.0	181.7	181.7	---
Floating Symbol Comparison	=F	4 to 6	6.6	8.3	---	---	---
	<>F						
	<F						
	<=F						
	>F						
	>=F						
FLOATING- POINT TO ASCII	FSTR	4	48.5	48.9	---	---	---
ASCII TO FLOATING-POINT	FVAL	3	21.1	21.3	---	---	---

4-3-14 Double-precision Floating-point Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
DOUBLE SYMBOL COMPARISON	=D	4	8.5	10.3	---	---	---
	<>D						
	<D						
	<=D						
	>D						
	>=D						
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	3	11.7	12.1	---	---	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	3	11.6	12.1	---	---	---
16-BIT BINARY TO DOUBLE FLOATING	DBL	3	9.9	10.0	---	---	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	3	9.8	10.0	---	---	---
DOUBLE FLOATING-POINT ADD	+D	4	11.2	11.9	---	---	---
DOUBLE FLOATING-POINT SUBTRACT	–D	4	11.2	11.9	---	---	---
DOUBLE FLOATING-POINT MULTIPLY	*D	4	12.0	12.7	---	---	---
DOUBLE FLOATING-POINT DIVIDE	/D	4	23.5	24.2	---	---	---
DOUBLE DEGREES TO RADIANS	RADD	3	27.4	27.8	---	---	---
DOUBLE RADIANS TO DEGREES	DEGD	3	11.2	11.9	---	---	---
DOUBLE SINE	SIND	3	45.4	45.8	---	---	---
DOUBLE COSINE	COSD	3	43.0	43.4	---	---	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
DOUBLE TANGENT	TAND	3	20.1	20.5	---	---	---
DOUBLE ARC SINE	ASIND	3	21.5	21.9	---	---	---
DOUBLE ARC COSINE	ACOSD	3	24.7	25.1	---	---	---
DOUBLE ARC TANGENT	ATAND	3	19.3	19.7	---	---	---
DOUBLE SQUARE ROOT	SQRTD	3	47.4	47.9	---	---	---
DOUBLE EXPONENT	EXPD	3	121.0	121.4	---	---	---
DOUBLE LOGARITHM	LOGD	3	16.0	16.4	---	---	---
DOUBLE EXPONENTIAL POWER	PWRD	4	223.9	224.2	---	---	---

4-3-15 Table Data Processing Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SET STACK	SSET	3	8.0	8.3	8.5	8.5	Designating 5 words in stack area
			231.6	251.8	276.8	276.8	Designating 1,000 words in stack area
PUSH ONTO STACK	PUSH	3	6.5	8.6	9.1	9.1	---
FIRST IN FIRST OUT	FIFO	3	6.9	8.9	10.6	10.6	Designating 5 words in stack area
			352.6	434.3	1130	1130	Designating 1,000 words in stack area
LAST IN FIRST OUT	LIFO	3	7.0	9.0	9.9	9.9	---
DIMENSION RECORD TABLE	DIM	5	15.2	21.6	142.1	142.1	---
SET RECORD LOCATION	SETR	4	5.4	5.9	7.0	7.0	---
GET RECORD NUMBER	GETR	4	7.8	8.4	11.0	11.0	---
DATA SEARCH	SRCH	4	15.5	19.5	19.5	19.5	Searching for 1 word
			2420	3340	3340	3340	Searching for 1,000 words[*1]
SWAP BYTES	SWAP	3	12.2	13.6	13.6	13.6	Swapping 1 word
			1940	2820	2820	2820	Swapping 1,000 words[*1]
FIND MAXIMUM	MAX	4 to 5	19.2	24.9	24.9	24.9	Searching for 1 word
			2390	3360	3360	3360	Searching for 1,000 words[*1]
FIND MINIMUM	MIN	4 to 5	19.2	25.3	25.3	25.3	Searching for 1 word
			2390	3330	3330	3330	Searching for 1,000 words [*1]
SUM	SUM	4 to 5	28.2	38.5	38.5	38.3	Adding 1 word
			1420	1950	1950	1950	Adding 1,000 words [*1]
FRAME CHECKSUM	FCS	4 to 5	20.0	28.3	28.3	28.3	For 1-word table length
			1650	2480	2480	2480	For 1,000-word table length[*1]
STACK SIZE READ	SNUM	3	6.0	6.3	---	---	---
STACK DATA READ	SREAD	4	8.0	8.4	---	---	---

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
STACK DATA OVERWRITE	SWRIT	4	7.2	7.6	---	---	---
STACK DATA INSERT	SINS	4	7.8	9.9	---	---	---
			354.0	434.8	---	---	For 1,000-word table
STACK DATA DELETE	SDEL	4	8.6	10.6	---	---	---
			354.0	436.0	---	---	For 1,000-word table

*1 The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-3-16 Data Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
PID CONTROL	PID	4	436.2	678.2	678.2	678.2	First execution
			332.3	474.9	474.9	474.9	Input ON and sampling
			97.3	141.3	141.3	141.3	Input ON and not sampling
PID CONTROL WITH AUTOTUNING	PIDAT	4	446.3	712.5	---	---	First execution
			339.4	533.9	---	---	Input ON and sampling
			100.7	147.1	---	---	Input ON and not sampling
			189.2	281.6	---	---	Initial execution of autotuning
			535.2	709.8	---	---	Autotuning when sampling
LIMIT CONTROL	LMT	4 to 5	16.1	22.1	22.1	22.1	---
DEAD BAND CONTROL	BAND	4 to 5	17.0	22.5	22.5	22.5	---
DEAD ZONE CONTROL	ZONE	4 to 5	15.4	20.5	20.5	20.5	---
TIME-PROPORTIONAL OUTPUT	TPO	4	10.4	14.8	---	---	Input OFF
			54.5	82.0	---	---	Input ON and duty specified or output limit disabled
			61.0	91.9	---	---	Input ON, manipulated variable specified, and output limit disabled
SCALING	SCL	4	37.1	53.0	56.8	56.8	---
SCALING 2	SCL2	4	28.5	40.2	50.7	50.7	---
SCALING 3	SCL3	4	33.4	47.0	57.7	57.7	---
AVERAGE	AVG	4	36.3	52.6	53.1	53.1	Average of an operation
			291.0	419.9	419.9	419.9	Average of 64 operations

4-3-17 Subroutine Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SUBROUTINE CALL	SBS	2	1.26	1.96	17.0	17.0	---
SUBROUTINE ENTRY	SBN	2	---	---	---	---	---
SUBROUTINE RETURN	RET	1	0.86	1.60	20.60	20.60	---
MACRO	MCRO	4	23.3	23.3	23.3	23.3	---
GLOBAL SUBROUTINE CALL	GSBN	2	---	---	---	---	---
GLOBAL SUBROUTINE ENTRY	GRET	1	1.26	1.96	---	---	---
GLOBAL SUBROUTINE RETURN	GSBS	2	0.86	1.60	---	---	---

4-3-18 Interrupt Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SET INTERRUPT MASK	MSKS	3	25.6	38.4	39.5	39.5	---
READ INTERRUPT MASK	MSKR	3	11.9	11.9	11.9	11.9	---
CLEAR INTERRUPT	CLI	3	27.4	41.3	41.3	41.3	---
DISABLE INTERRUPTS	DI	1	15.0	16.8	16.8	16.8	---
ENABLE INTERRUPTS	EI	1	19.5	21.8	21.8	21.8	---

4-3-19 Step Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
STEP DEFINE	STEP	2	17.4	20.7	27.1	27.1	Step control bit ON
			11.8	13.7	24.4	24.4	Step control bit OFF
STEP START	SNXT	2	6.6	7.3	10.0	10.0	---

4-3-20 Basic I/O Unit Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
I/O REFRESH	IORF	3	15.5	16.4	23.5	23.5	For 1 word, CS-series Input Unit
			17.20	18.40	25.6	25.6	For 1 word, CS-series Output Unit
			58.5	63.2	81.7	81.7	For 1 word, C200H Input Unit
			62.6	67.0	86.7	86.7	For 1 word, C200H Output Unit
			319.9	320.7	377.5	377.6	For 60 words, CS-series Input Unit
			358.0	354.4	460.1	460.1	For 60 words, CS-series Output Unit
			303.3	343.9	357.1	357.1	For 10 words, C200H Input Unit
			348.2	376.6	407.5	407.5	For 10 words, C200H Output Unit

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
7-SEGMENT DECODER	SDEC	4	6.5	6.9	14.1	14.1	---
DIGITAL SWITCH INPUT	DSW	6	50.7	73.5	---	---	4 digits, data input value: 0
			51.3	73.5	---	---	8 digits, data input value: 00
TEN KEY INPUT	TKY	4	9.7	13.2	---	---	Data input value: 00
			10.7	14.8	---	---	Data input value: FF
HEXADECIMAL KEY INPUT	HKY	5	50.3	70.9	---	---	Data input value: 00
			50.1	71.2	---	---	Data input value: FF
MATRIX INPUT	MTR	5	47.8	68.1	---	---	Data input value: 00
			48.0	68.0	---	---	Data input value: FF
7-SEGMENT DISPLAY OUTPUT	7SEG	5	58.1	83.3	---	---	4 digits
			63.3	90.3	---	---	8 digits
INTELLIGENT I/O READ	IORD	4	*1	*1	*1	*1	---
INTELLIGENT I/O WRITE	IOWR	4	*1	*1	*1	*1	---
CPU BUS I/O REFRESH	DLNK	4	287.8	315.5	---	---	Allocated 1 word

*1 Execution of the IORD and IORW instructions depends on the Special I/O Units for which they are being executed.

4-3-21 Serial Communications Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
PROTOCOL MACRO	PMCR	5	100.1	142.1	276.8	276.8	Direct specification
			134.2	189.6	305.9	305.9	Operand specification, sending 1 word, receiving 1 word
TRANSMIT	TXD	4	68.5	98.8	98.8	98.8	Sending 1 byte
			734.3	1100	1100	1100	Sending 256 bytes
RECEIVE	RXD	4	89.6	131.1	131.1	131.1	Storing 1 byte
			724.2	1110	1110	1110	Storing 256 bytes
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	4	131.5	202.4	---	---	Sending 1 byte
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	4	131	200.8	---	---	Storing 1 byte
CHANGE SERIAL PORT SETUP	STUP	3	341.2	400.0	440.4	440.4	---

4-3-22 Network Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
NETWORK SEND	SEND	4	84.4	123.9	123.9	123.9	---
NETWORK RECEIVE	RECV	4	85.4	124.7	124.7	124.7	---
DELIVER COMMAND	CMND	4	106.8	136.8	136.8	136.8	---
EXPLICIT MESSAGE SEND	EXPLT	4	127.6	190.0	---	---	---
EXPLICIT GET ATTRIBUTE	EGATR	4	123.9	185.0	---	---	---
EXPLICIT SET ATTRIBUTE	ESATR	3	110.0	164.4	---	---	---
EXPLICIT WORD READ	ECHRD	4	106.8	158.9	---	---	---
EXPLICIT WORD WRITE	ECHWR	4	106.0	158.3	---	---	---

4-3-23 File Memory Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
READ DATA FILE	FREAD	5	391.4	632.4	684.1	684.1	2-character directory + file name in binary
WRITE DATA FILE	FWRIT	5	387.8	627.0	684.7	684.7	2-character directory + file name in binary
WRITE TEXT FILE	TWRIT	5	390.1	619.1	---	---	

4-3-24 Display Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
DISPLAY MESSAGE	MSG	3	10.1	14.2	14.3	14.3	Displaying message
			8.4	11.3	11.3	11.3	Deleting displayed message

4-3-25 Clock Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
CALENDAR ADD	CADD	4	38.3	201.9	209.5	209.5	---
CALENDAR SUBTRACT	CSUB	4	38.6	170.4	184.1	184.1	---
HOURS TO SECONDS	SEC	3	21.4	29.3	35.8	35.8	---
SECONDS TO HOURS	HMS	3	22.2	30.9	42.1	42.1	---
CLOCK ADJUSTMENT	DATE	2	60.5	87.4	95.9	95.9	---

4-3-26 Debugging Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (μs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
Trace Memory Sampling	TRSM	1	80.4	120.0	120.0	120.0	Sampling 1 bit and 0 words
			848.1	1060	1060	1060	Sampling 31 bits and 6 words

4-3-27 Failure Diagnosis Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
FAILURE ALARM	FAL	3	15.4	16.7	16.7	16.7	Recording errors
			179.8	244.8	244.8	244.8	Deleting errors (in order of priority)
			432.4	657.1	657.1	657.1	Deleting errors (all errors)
			161.5	219.4	219.4	219.4	Deleting errors (individually)
SEVERE FAILURE ALARM	FALS	3	---	---	---	---	---
FAILURE POINT DETECTION	FPD	4	140.9	202.3	202.3	202.3	Bit address output, time monitored
			163.4	217.6	217.6	217.6	Bit address output, first error detection
			185.2	268.9	268.9	268.9	Message characters output, time monitored
			207.5	283.6	283.6	283.6	Message characters output, first error detection

4-3-28 Other Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
SET CARRY	STC	1	0.06	0.06	0.12	0.12	---
CLEAR CARRY	CLC	1	0.06	0.06	0.12	0.12	---
SELECT EM BANK	EMBC	2	14.0	15.1	15.1	15.1	---
EXTEND MAXIMUM CYCLE TIME	WDT	2	15.0	19.7	19.7	19.7	---
SAVE CONDITION FLAGS	CCS	1	8.6	12.5	---	---	---
LOAD CONDITION FLAGS	CCL	1	9.8	13.9	---	---	---
CONVERT ADDRESS FROM CV	FRMCV	3	13.6	19.9	---	---	---
CONVERT ADDRESS TO CV	TOCV	3	11.9	17.2	---	---	---
DISABLE PERIPHERAL SERVICING	IOSP	---	13.9	19.8	---	---	---
ENABLE PERIPHERAL SERVICING	IORS	---	63.6	92.3	---	---	---

4-3-29 Block Programming Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
BLOCK PROGRAM BEGIN	BPRG	2	12.1	13.0	13.0	13.0	---
BLOCK PROGRAM END	BEND	1	9.6	12.3	13.1	13.1	---
BLOCK PROGRAM PAUSE	BPPS	2	10.6	12.3	14.9	14.9	---
BLOCK PROGRAM RESTART	BPRS	2	5.1	5.6	8.3	8.3	---

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
CONDITIONAL BLOCK EXIT	(Execution condition) EXIT	1	10.0	11.3	12.9	12.9	Block exited (input condition ON)
			4.0	4.9	7.3	7.3	Block not exited (input condition OFF)
CONDITIONAL BLOCK EXIT	EXIT (bit address)	2	6.8	13.5	16.3	16.3	Block exited (bit ON)
			4.7	7.2	10.7	10.7	Block not exited (bit OFF)
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT (bit address)	2	12.4	14.0	16.8	16.8	Block exited (bit OFF)
			7.1	7.6	11.2	11.2	Block not exited (bit ON)
Branching	IF (execution condition)	1	4.6	4.8	7.2	7.2	IF true (input condition ON)
			6.7	7.3	10.9	10.9	IF false (input condition OFF)
Branching	IF (bit address)	2	6.8	7.2	10.4	10.4	IF true (bit ON)
			9.0	9.6	14.2	14.2	IF false (bit OFF)
Branching (NOT)	IF NOT (bit address)	2	7.1	7.6	10.9	10.9	IF true (bit OFF)
			9.2	10.1	14.7	14.7	IF false (bit ON)
Branching	ELSE	1	6.2	6.7	9.9	9.9	IF true
			6.8	7.7	11.2	11.2	IF false
Branching	IEND	1	6.9	7.7	11.0	11.0	IF true
			4.4	4.6	7.0	7.0	IF false
ONE CYCLE AND WAIT	WAIT (execution condition)	1	12.6	13.7	16.7	16.7	Do not wait (input condition ON)
			3.9	4.1	6.3	6.3	Wait (input condition OFF)
ONE CYCLE AND WAIT	WAIT (bit address)	2	12.0	13.4	16.5	16.5	Do not wait (bit ON)
			6.1	6.5	9.6	9.6	Wait (bit OFF)
ONE CYCLE AND WAIT (NOT)	WAIT NOT (bit address)	2	12.2	13.8	17.0	17.0	Do not wait (bit OFF)
			6.4	6.9	10.1	10.1	Wait (bit ON)
HUNDRED-MS TIMER WAIT	TIMW	3	22.3	25.2	33.1	33.1	Default setting
			24.9	27.8	35.7	35.7	Normal execution
	TIMWX	3	22.3	25.2	33.1	33.1	Default setting
			24.9	27.8	35.7	35.7	Normal execution
TEN-MS TIMER WAIT	TMHW	3	25.8	27.9	34.1	34.1	Default setting
			20.6	22.7	28.9	28.9	Normal execution
	TMHWX	3	25.8	27.9	34.1	34.1	Default setting
			20.6	22.7	28.9	28.9	Normal execution
COUNTER WAIT	CNTW	4	17.9	22.6	27.4	27.4	Default setting
			19.1	23.9	28.7	28.7	Normal execution
	CNTWX	4	17.9	22.6	27.4	27.4	Default setting
			19.1	23.9	28.7	28.7	Normal execution

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
Loop Control	LOOP	1	7.9	9.1	12.3	12.3	---
Loop Control	LEND (execution condition)	1	7.7	8.4	10.9	10.9	Do not loop (input condition ON)
			6.8	8.0	9.8	9.8	Loop (input condition OFF)
Loop Control	LEND (bit address)	2	9.9	10.7	14.4	14.4	Do not loop (bit ON)
			8.9	10.3	13.0	13.0	Loop (bit OFF)
Loop Control	LEND NOT (bit address)	2	10.2	11.2	14.8	14.8	Do not loop (bit OFF)
			9.3	10.8	13.5	13.5	Loop (bit ON)

4-3-30 Text String Processing Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
MOV STRING	MOV$	3	45.6	66.0	84.3	84.3	Transferring 1 character[1]
CONCATENATE STRING	+$	4	86.5	126.0	167.8	167.8	1 character + 1 character[1]
GET STRING LEFT	LEFT$	4	53.0	77.4	94.3	94.3	Retrieving 1 character from 2 characters[1]
GET STRING RIGHT	RGHT$	4	52.2	76.3	94.2	94.2	Retrieving 1 character from 2 characters[1]
GET STRING MIDDLE	MID$	5	56.5	84.6	230.2	230.2	Retrieving 1 character from 3 characters[1]
FIND IN STRING	FIND$	4	51.4	77.5	94.1	94.1	Searching for 1 character from 2 characters[1]
STRING LENGTH	LEN$	3	19.8	28.9	33.4	33.4	Detecting 1 character [1]
REPLACE IN STRING	RPLC$	6	175.1	258.7	479.5	479.5	Replacing the first of 2 characters with 1 character[1]
DELETE STRING	DEL$	5	63.4	94.2	244.6	244.6	Deleting the leading character of 2 characters [1]
EXCHANGE STRING	XCHG$	3	60.6	87.2	99.0	99.0	Exchanging 1 character with 1 character [1]
CLEAR STRING	CLR$	2	23.8	36.0	37.8	37.8	Clearing 1 character[1]
INSERT INTO STRING	INS$	5	136.5	200.6	428.9	428.9	Inserting 1 character after the first of 2 characters[1]
String Comparison Instructions	=$	4	48.5	69.8	86.2	86.2	Comparing 1 character with 1 character
	<>$						
	<$						
	+>$						
	+>=$						

[1] The instruction execution time is greatly affected by the amount to data. This will affect the cycle time. To reduce the effect on the cycle time, background execution can be specified.

4-3-31 Task Control Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
TASK ON	TKON	2	6.4	10.2	3.2	3.2	Cyclic task specified
			20.6	30.1	---	---	Extra task specified
TASK OFF	TKOF	2	123.7	195.8	61.3	61.3	Cyclic task specified
			23.4	34.1	---	---	Extra task specified

4-3-32 Model Conversion Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
BLOCK TRANSFER	XFERC	4	6.4	6.5	---	---	Transferring 1 word
			481.6	791.6	---	---	Transferring 1,000 words
SINGLE WORD DISTRIBUTE	DISTC	4	3.4	3.5	---	---	Data distribute
			5.9	7.3	---	---	Stack operation
DATA COLLECT	COLLC	4	3.5	3.85	---	---	Data distribute
			8	9.1	---	---	Stack operation
			8.3	9.6	---	---	Stack operation 1 word FIFO Read
			2052.3	2097.5	---	---	Stack operation 1,000 word FIFO Read
MOVE BIT	MOVBC	4	4.5	4.88	---	---	---
BIT COUNTER	BCNTC	4	4.9	5	---	---	Counting 1 word
			1252.4	1284.4	---	---	Counting 1,000 words

4-3-33 Special Function Block Instructions

Instruction	Mnemonic	Length (steps)	ON execution time (µs)				Conditions
			CS1H CPU6□H	CS1G CPU4□H	CS1H CPU6□	CS1G CPU4□	
GET VARIABLE ID	GETID	4	14	22.2	---	---	---

4-3-34 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 using CS1-H CPU Units with unit version 3.0 or later and CS1D CPU Units with unit version 4.0 or later.

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			CPU Unit model			
			CS1H-CPU6□H	**CS1G-CPU4□H**	**CS1D-CPU□□HA**	**CS1D-CPU□□SA**
A	Startup time	Startup time not including I/O parameter transfer	6.8 µs	8.8 µs	8.0 µs	6.9 µs
B	I/O parameter transfer processing time The data type is indicated in parentheses.	1-bit (BOOL) input symbol or output symbol	0.4 µs	0.7 µs	0.6 µs	0.5 µs
		1-word (INT, UINT, WORD) input symbol or output symbol	0.3 µs	0.6 µs	0.5 µs	0.4 µs
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	0.5 µs	0.8 µs	0.7 µs	0.7 µs
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	1.0 µs	1.6 µs	2.4 µs	2.2 µs
		I/O symbols	0.4 µs	0.5 µs	0.7 µs	0.6 µs
C	Function block definition instruction execution time	Total instruction processing time (same as standard user program)				

Example: CS1H-CPU63H
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.

Note Guidelines on Converting Program Capacities from Previous OMRON PLCs

Guidelines are provided in the following table for converting the program capacity (unit: words) of previous OMRON PLCs (SYSMAC C200HX/HG/HE, CVM1, or CV-series PLCs) to the program capacity (unit: steps) of the CS-series PLCs.

Add the following value (n) to the program capacity (unit: words) of the previous PLCs for instruction to obtain the program capacity (unit: steps) of the CS-series PLCs.

CS-series steps = "a" (words) of previous PLC + n			
Instructions	**Variations**	**Value of n when converting from C200HX/HG/HE to CS Series**	**Value of n when converting from CV-series PLC or CVM1 to CS Series**
Basic instructions	None	OUT, SET, RSET, or KEEP(011): –1 Other instructions: 0	0
	Upward Differentiation	None	+1
	Immediate Refreshing	None	0
	Upward Differentiation and Immediate Refreshing	None	+2
Special instructions	None	0	–1
	Upward Differentiation	+1	0
	Immediate Refreshing	None	+3
	Upward Differentiation and Immediate Refreshing	None	+4

For example, if OUT is used with an address of CIO 000000 to CIO 25515, the program capacity of a C200HX/HG/HE PLC would be 2 words per instruction and that of the CS-series PLC would be 1 (2 – 1) step per instruction.

For example, if !MOV is used (MOVE instruction with immediate refreshing), the program capacity of a CVM1 or CV-series PLC would be 4 words per instruction and that of the CS-series PLC would be 7 (4 + 3) steps.

Note Number of Function Block Program Steps (CS1-H CPU Units with unit version 3.0 or later and CS1D CPU Units with unit version 4.0 or later)

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 using CS1-H CPU Units with unit version 3.0 or later and CS1D CPU Units with unit version 4.0 or later.

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			CS1-H CPU Units with unit version 3.0 or later and CS1D CPU Units with unit version 4.0 or later
m	Call part		57 steps
n	I/O parameter transfer part The data type is shown in parentheses.	1-bit (BOOL) input symbol or output symbol	6 steps
		1-word (INT, UINT, WORD) input symbol or output symbol	6 steps
		2-word (DINT, UDINT, DWORD, REAL) input symbol or output symbol	6 steps
		4-word (LINT, ULINT, LWORD, LREAL) input symbol or output symbol	18 steps
		I/O symbols	6 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
List of Instructions by Function Code

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
---	LOAD	LD	@LD	%LD	!LD	56
---	LOAD NOT	LD NOT	---	---	!LD NOT	56
---	AND	AND	@AND	%AND	!AND	58
---	AND NOT	AND NOT	---	---	!AND NOT	58
---	OR	OR	@OR	%OR	!OR	60
---	OR NOT	OR NOT	---	---	!OR NOT	60
---	AND LOAD	AND LD	---	---	---	62
---	OR LOAD	OR LD	---	---	---	62
---	OUTPUT	OUT	---	---	!OUT	74
---	OUTPUT NOT	OUT NOT	---	---	!OUT NOT	74
---	SET	SET	@SET	%SET	!SET	86
---	RESET	RSET	@RSET	%RSET	!RSET	86
---	HUNDRED-MS TIMER	TIM	---	---	---	133
---	COUNTER	CNT	---	---	---	160
000	NO OPERATION	NOP	---	---	---	98
001	END	END	---	---	---	97
002	INTERLOCK	IL	---	---	---	99
003	INTERLOCK CLEAR	ILC	---	---	---	99
004	JUMP	JMP	---	---	---	112
005	JUMP END	JME	---	---	---	112
006	FAILURE ALARM	FAL	@FAL	---	---	944
007	SEVERE FAILURE ALARM	FALS	---	---	---	951
008	STEP DEFINE	STEP	---	---	---	702
009	STEP START	SNXT	---	---	---	702
010	SHIFT REGISTER	SFT	---	---	---	223
011	KEEP	KEEP	---	---	!KEEP	78
012	REVERSIBLE COUNTER	CNTR	---	---	---	163
013	DIFFERENTIATE UP	DIFU	---	---	!DIFU	82
014	DIFFERENTIATE DOWN	DIFD	---	---	!DIFD	84
015	TEN-MS TIMER	TIMH	---	---	---	137
016	WORD SHIFT	WSFT	@WSFT	---	---	229
017	ASYNCHRONOUS SHIFT REGISTER	ASFT	@ASFT	---	---	227
019	MULTIPLE COMPARE	MCMP	@MCMP	---	---	183
020	COMPARE	CMP	---	---	!CMP	177
021	MOVE	MOV	@MOV	---	!MOV	199
022	MOVE NOT	MVN	@MVN	---	---	202
023	BCD TO BINARY	BIN	@BIN	---	---	298
024	BINARY TO BCD	BCD	@BCD	---	---	300
025	ARITHMETIC SHIFT LEFT	ASL	@ASL	---	---	231
026	ARITHMETIC SHIFT RIGHT	ASR	@ASR	---	---	233
027	ROTATE LEFT	ROL	@ROL	---	---	235
028	ROTATE RIGHT	ROR	@ROR	---	---	239
029	COMPLEMENT	COM	---	---	---	362
034	LOGICAL AND	ANDW	@ANDW	---	---	354
035	LOGICAL OR	ORW	@ORW	---	---	356

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
036	EXCLUSIVE OR	XORW	@XORW	---	---	358
037	EXCLUSIVE NOR	XNRW	@XNRW	---	---	360
040	SET CARRY	STC	@STC	---	---	965
041	CLEAR CARRY	CLC	@CLC	---	---	965
045	TRACE MEMORY SAMPLING	TRSM	---	---	---	941
046	DISPLAY MESSAGE	MSG	@MSG	---	---	927
058	DOUBLE BCD TO DOUBLE BINARY	BINL	@BINL	---	---	298
059	DOUBLE BINARY TO BCD	BCDL	@BCDL	---	---	300
060	DOUBLE COMPARE	CMPL	---	---	---	177
062	MULTIPLE BIT TRANSFER	XFRB	@XFRB	---	---	208
063	COLUMN TO LINE	LINE	@LINE	---	---	326
064	LINE TO COLUMN	COLM	@COLM	---	---	328
065	HOURS TO SECONDS	SEC	@SEC	---	---	935
066	SECONDS TO HOURS	HMS	@HMS	---	---	937
067	BIT COUNTER	BCNT	@BCNT	---	---	381
068	UNSIGNED BLOCK COMPARE	BCMP	@BCMP	---	---	187
069	ARITHMETIC PROCESS	APR	@APR	---	---	369
070	BLOCK TRANSFER	XFER	@XFER	---	---	210
071	BLOCK SET	BSET	@BSET	---	---	212
072	BCD SQUARE ROOT	ROOT	@ROOT	---	---	366
073	DATA EXCHANGE	XCHG	@XCHG	---	---	214
074	ONE DIGIT SHIFT LEFT	SLD	@SLD	---	---	243
075	ONE DIGIT SHIFT RIGHT	SRD	@SRD	---	---	243
076	DATA DECODER	MLPX	@MLPX	---	---	307
077	DATA ENCODER	DMPX	@DMPX	---	---	312
078	7-SEGMENT DECODER	SDEC	@SDEC	---	---	722
079	FLOATING POINT DIVIDE	FDIV	@FDIV	---	---	378
080	SINGLE WORD DISTRIBUTE	DIST	@DIST	---	---	216
081	DATA COLLECT	COLL	@COLL	---	---	218
082	MOVE BIT	MOVB	@MOVB	---	---	204
083	MOVE DIGIT	MOVD	@MOVD	---	---	206
084	REVERSIBLE SHIFT REGISTER	SFTR	@SFTR	---	---	225
085	TABLE COMPARE	TCMP	@TCMP	---	---	185
086	ASCII CONVERT	ASC	@ASC	---	---	317
087	ACCUMULATIVE TIMER	TTIM	---	---	---	150
088	AREA RANGE COMPARE	ZCP	---	---	---	192
090	NETWORK SEND	SEND	@SEND	---	---	851
091	SUBROUTINE CALL	SBS	@SBS	---	---	604
092	SUBROUTINE ENTRY	SBN	---	---	---	613
093	SUBROUTINE RETURN	RET	---	---	---	613
094	EXTEND MAXIMUM CYCLE TIME	WDT	@WDT	---	---	968
096	BLOCK PROGRAM BEGIN	BPRG	---	---	---	986
097	I/O REFRESH	IORF	@IORF	---	---	712

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
098	NETWORK RECEIVE	RECV	@RECV	---	---	861
099	MACRO	MCRO	@MCRO	---	---	610
114	SIGNED BINARY COMPARE	CPS	---	---	!CPS	180
115	DOUBLE SIGNED BINARY COMPARE	CPSL	---	---	---	180
116	DOUBLE AREA RANGE COMPARE	ZCPL	---	---	---	192
117	SIGNED AREA RANGE COMPARE	ZCPS	---	---	---	196
118	DOUBLE SIGNED AREA RANGE COMPARE	ZCPSL	---	---	---	196
160	2'S COMPLEMENT	NEG	@NEG	---	---	303
161	DOUBLE 2'S COMPLEMENT	NEGL	@NEGL	---	---	303
162	ASCII TO HEX	HEX	@HEX	---	---	321
174	DOUBLE FIND MAXIMUM	MAXL	@MAXL	---	---	508
175	DOUBLE FIND MINIMUM	MINL	@MINL	---	---	515
176	FIND MAXIMUM FLOATING	MAXF	@MAXF	---	---	511
177	FIND MINIMUM FLOATING	MINF	@MINF	---	---	517
178	FIND DOUBLE MAXIMUM FLOATING	MAXD	@MAXD	---	---	513
179	FIND DOUBLE MINIMUM FLOATING	MIND	@MIND	---	---	519
180	FRAME CHECKSUM	FCS	@FCS	---	---	524
181	DATA SEARCH	SRCH	@SRCH	---	---	499
182	FIND MAXIMUM	MAX	@MAX	---	---	504
183	FIND MINIMUM	MIN	@MIN	---	---	504
184	SUM	SUM	@SUM	---	---	521
190	PID CONTROL	PID	---	---	---	556
191	PID CONTROL WITH AUTOTUNING	PIDAT	---	---	---	567
194	SCALING	SCL	@SCL	---	---	589
195	AVERAGE	AVG	---	---	---	600
203	UNSIGNED ONE-WORD RECORD SORT	RSORT	@RSORT	---	---	546
204	UNSIGNED TWO-WORD RECORD SORT	RSORT2	@RSORT2	---	---	550
205	UNSIGNED FOUR-WORD RECORD SORT	RSORT4	@RSORT4	---	---	553
210	DIGITAL SWITCH INPUT	DSW	---	---	---	725
211	TEN KEY INPUT	TKY	@TKY	---	---	729
212	HEXADECIMAL KEY INPUT	HKY	---	---	---	732
213	MATRIX INPUT	MTR	---	---	---	736
214	7-SEGMENT DISPLAY OUTPUT	7SEG	---	---	---	740
216	ANALOG INPUT DIRECT CONVERSION	AIDC	@AIDC	---	---	744
217	ANALOG OUTPUT DIRECT CONVERSION	AODC	@AODC	---	---	747
218	PCU HIGH-SPEED POSITIONING	NCDMV	@NCDMV	---	---	750

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
219	PCU POSITIONING TRIGGER	NCDTR	@NCDTR	---	---	754
222	INTELLIGENT I/O READ	IORD	@IORD	---	---	757
223	INTELLIGENT I/O WRITE	IOWR	@IOWR	---	---	760
225	SPECIAL I/O UNIT I/O REFRESH	FIORF	@FIORF	---	---	715
226	CPU BUS UNIT I/O REFRESH	DLNK	@DLNK	---	---	718
235	RECEIVE	RXD	@RXD	---	---	779
236	TRANSMIT	TXD	@TXD	---	---	773
237	CHANGE SERIAL PORT SETUP	STUP	@STUP	---	---	817
255	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	@RXDU	---	---	796
256	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	@TXDU	---	---	789
260	PROTOCOL MACRO	PMCR	@PMCR	---	---	766
261	DIRECT RECEIVE VIA SERIAL COMMUNICATIONS UNIT	DRXDU	@DRXDU	---	---	809
262	DIRECT TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	DTXDU	@DTXDU	---	---	803
264	PROTOCOL MACRO 2	PMCR2	@PMCR2	---	---	821
269	FAILURE POINT DETECTION	FPD	---	---	---	957
281	SELECT EM BANK	EMBC	@EMBC	---	---	966
282	SAVE CONDITION FLAGS	CCS	@CCS	---	---	970
283	LOAD CONDITION FLAGS	CCL	@CCL	---	---	970
284	CONVERT ADDRESS FROM CV	FRMCV	@FRMCV	---	---	973
285	CONVERT ADDRESS TO CV	TOCV	@TOCV	---	---	977
286	GET VARIABLE ID	GETID	@GETID	---	---	1056
287	DISABLE PERIPHERAL SERVICING	IOSP	@IOSP	---	---	980
288	ENABLE PERIPHERAL SERVICING	IORS	---	---	---	980
300	AND EQUAL	AND =	---	---	---	169
300	LOAD EQUAL	LD =	---	---	---	169
300	OR EQUAL	OR =	---	---	---	169
301	AND DOUBLE EQUAL	AND =L	---	---	---	169
301	LOAD DOUBLE EQUAL	LD =L	---	---	---	169
301	OR DOUBLE EQUAL	OR =L	---	---	---	169
302	AND SIGNED EQUAL	AND =S	---	---	---	169
302	LOAD SIGNED EQUAL	LD =S	---	---	---	169
302	OR SIGNED EQUAL	OR =S	---	---	---	169
303	AND DOUBLE SIGNED EQUAL	AND =SL	---	---	---	169
303	LOAD DOUBLE SIGNED EQUAL	LD =SL	---	---	---	169
303	OR DOUBLE SIGNED EQUAL	OR =SL	---	---	---	169
305	AND NOT EQUAL	AND <>	---	---	---	169
305	LOAD NOT EQUAL	LD <>	---	---	---	169
305	OR NOT EQUAL	OR <>	---	---	---	169

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
306	AND DOUBLE NOT EQUAL	AND <>L	---	---	---	169
306	LOAD DOUBLE NOT EQUAL	LD <>L	---	---	---	169
306	OR DOUBLE NOT EQUAL	OR <>L	---	---	---	169
307	AND SIGNED NOT EQUAL	AND <>S	---	---	---	169
307	LOAD SIGNED NOT EQUAL	LD <>S	---	---	---	169
307	OR SIGNED NOT EQUAL	OR <>S	---	---	---	169
308	AND DOUBLE SIGNED NOT EQUAL	AND <>SL	---	---	---	169
308	LOAD DOUBLE SIGNED NOT EQUAL	LD <>SL	---	---	---	169
308	OR DOUBLE SIGNED NOT EQUAL	OR <>SL	---	---	---	169
310	AND LESS THAN	AND <	---	---	---	169
310	LOAD LESS THAN	LD <	---	---	---	169
310	OR LESS THAN	OR <	---	---	---	169
311	AND DOUBLE LESS THAN	AND <L	---	---	---	169
311	LOAD DOUBLE LESS THAN	LD <L	---	---	---	169
311	OR DOUBLE LESS THAN	OR <L	---	---	---	169
312	AND SIGNED LESS THAN	AND <S	---	---	---	169
312	LOAD SIGNED LESS THAN	LD <S	---	---	---	169
312	OR SIGNED LESS THAN	OR <S	---	---	---	169
313	AND DOUBLE SIGNED LESS THAN	AND <SL	---	---	---	169
313	LOAD DOUBLE SIGNED LESS THAN	LD <SL	---	---	---	169
313	OR DOUBLE SIGNED LESS THAN	OR <SL	---	---	---	169
315	AND LESS THAN OR EQUAL	AND <=	---	---	---	169
315	LOAD LESS THAN OR EQUAL	LD <=	---	---	---	169
315	OR LESS THAN OR EQUAL	OR <=	---	---	---	169
316	AND DOUBLE LESS THAN OR EQUAL	AND <=L	---	---	---	169
316	LOAD DOUBLE LESS THAN OR EQUAL	LD <=L	---	---	---	169
316	OR DOUBLE LESS THAN OR EQUAL	OR <=L	---	---	---	169
317	AND SIGNED LESS THAN OR EQUAL	AND <=S	---	---	---	169
317	LOAD SIGNED LESS THAN OR EQUAL	LD <=S	---	---	---	169
317	OR SIGNED LESS THAN OR EQUAL	OR <=S	---	---	---	169
318	AND DOUBLE SIGNED LESS THAN OR EQUAL	AND <=SL	---	---	---	169
318	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	LD <=SL	---	---	---	169
318	OR DOUBLE SIGNED LESS THAN OR EQUAL	OR <=SL	---	---	---	169

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
320	AND GREATER THAN	AND >	---	---	---	169
320	LOAD GREATER THAN	LD >	---	---	---	169
320	OR GREATER THAN	OR >	---	---	---	169
321	AND DOUBLE GREATER THAN	AND >L	---	---	---	169
321	LOAD DOUBLE GREATER THAN	LD >L	---	---	---	169
321	OR DOUBLE GREATER THAN	OR >L	---	---	---	169
322	AND SIGNED GREATER THAN	AND >S	---	---	---	169
322	LOAD SIGNED GREATER THAN	LD >S	---	---	---	169
322	OR SIGNED GREATER THAN	OR >S	---	---	---	169
323	AND DOUBLE SIGNED GREATER THAN	AND >SL	---	---	---	169
323	LOAD DOUBLE SIGNED GREATER THAN	LD >SL	---	---	---	169
323	OR DOUBLE SIGNED GREATER THAN	OR >SL	---	---	---	169
325	AND GREATER THAN OR EQUAL	AND >=	---	---	---	169
325	LOAD GREATER THAN OR EQUAL	LD >=	---	---	---	169
325	OR GREATER THAN OR EQUAL	OR >=	---	---	---	169
326	AND DOUBLE GREATER THAN OR EQUAL	AND >=L	---	---	---	169
326	LOAD DOUBLE GREATER THAN OR EQUAL	LD >=L	---	---	---	169
326	OR DOUBLE GREATER THAN OR EQUAL	OR >=L	---	---	---	169
327	AND SIGNED GREATER THAN OR EQUAL	AND >=S	---	---	---	169
327	LOAD SIGNED GREATER THAN OR EQUAL	LD >=S	---	---	---	169
327	OR SIGNED GREATER THAN OR EQUAL	OR >=S	---	---	---	169
328	AND DOUBLE SIGNED GREATER THAN OR EQUAL	AND >=SL	---	---	---	169
328	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	LD >=SL	---	---	---	169
328	OR DOUBLE SIGNED GREATER THAN OR EQUAL	OR >=SL	---	---	---	169
329	AND FLOATING EQUAL	AND =F	---	---	---	418
329	LOAD FLOATING EQUAL	LD =F	---	---	---	418
329	OR FLOATING EQUAL	OR =F	---	---	---	418
330	AND FLOATING NOT EQUAL	AND <>F	---	---	---	418
330	LOAD FLOATING NOT EQUAL	LD <>F	---	---	---	418
330	OR FLOATING NOT EQUAL	OR <>F	---	---	---	418

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
331	AND FLOATING LESS THAN	AND <F	---	---	---	418
331	LOAD FLOATING LESS THAN	LD <F	---	---	---	418
331	OR FLOATING LESS THAN	OR <F	---	---	---	418
332	AND FLOATING LESS THAN OR EQUAL	AND <=F	---	---	---	418
332	LOAD FLOATING LESS THAN OR EQUAL	LD <=F	---	---	---	418
332	OR FLOATING LESS THAN OR EQUAL	OR <=F	---	---	---	418
333	AND FLOATING GREATER THAN	AND >F	---	---	---	418
333	LOAD FLOATING GREATER THAN	LD >F	---	---	---	418
333	OR FLOATING GREATER THAN	OR >F	---	---	---	418
334	AND FLOATING GREATER THAN OR EQUAL	AND >=F	---	---	---	418
334	LOAD FLOATING GREATER THAN OR EQUAL	LD >=F	---	---	---	418
334	OR FLOATING GREATER THAN OR EQUAL	OR >=F	---	---	---	418
335	AND DOUBLE FLOATING EQUAL	AND =D	---	---	---	462
335	LOAD DOUBLE FLOATING EQUAL	LD =D	---	---	---	462
335	OR DOUBLE FLOATING EQUAL	OR =D	---	---	---	462
336	AND DOUBLE FLOATING NOT EQUAL	AND <>D	---	---	---	462
336	LOAD DOUBLE FLOATING NOT EQUAL	LD <>D	---	---	---	462
336	OR DOUBLE FLOATING NOT EQUAL	OR <>D	---	---	---	462
337	AND DOUBLE FLOATING LESS THAN	AND <D	---	---	---	462
337	LOAD DOUBLE FLOATING LESS THAN	LD <D	---	---	---	462
337	OR DOUBLE FLOATING LESS THAN	OR <D	---	---	---	462
338	AND DOUBLE FLOATING LESS THAN OR EQUAL	AND <=D	---	---	---	462
338	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	LD <=D	---	---	---	462
338	OR DOUBLE FLOATING LESS THAN OR EQUAL	OR <=D	---	---	---	462
339	AND DOUBLE FLOATING GREATER THAN	AND >D	---	---	---	462
339	LOAD DOUBLE FLOATING GREATER THAN	LD >D	---	---	---	462
339	OR DOUBLE FLOATING GREATER THAN	OR >D	---	---	---	462
340	AND DOUBLE FLOATING GREATER THAN OR EQUAL	AND >=D	---	---	---	462

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
340	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	LD >=D	---	---	---	462
340	OR DOUBLE FLOATING GREATER THAN OR EQUAL	OR >=D	---	---	---	462
341	AND TIME EQUAL	AND =DT	---	---	---	173
341	LOAD TIME EQUAL	LD =DT	---	---	---	173
341	OR TIME EQUAL	OR =DT	---	---	---	173
342	AND TIME NOT EQUAL	AND <> DT	---	---	---	173
342	LOAD TIME NOT EQUAL	LD <>DT	---	---	---	173
342	OR TIME NOT EQUAL	OR <>DT	---	---	---	173
343	AND TIME LESS THAN	AND <DT	---	---	---	173
343	LOAD TIME LESS THAN	LD <DT	---	---	---	173
343	OR TIME LESS THAN	OR <DT	---	---	---	173
344	AND TIME LESS THAN OR EQUAL	AND <=DT	---	---	---	173
344	LOAD TIME LESS THAN OR EQUAL	LD <=DT	---	---	---	173
344	OR TIME LESS THAN OR EQUAL	OR <=DT	---	---	---	173
345	AND TIME GREATER THAN	AND >DT	---	---	---	173
345	LOAD TIME GREATER THAN	LD >DT	---	---	---	173
345	OR TIME GREATER THAN	OR >DT	---	---	---	173
346	AND TIME GREATER THAN OR EQUAL	AND >=DT	---	---	---	173
346	LOAD TIME GREATER THAN OR EQUAL	LD >=DT	---	---	---	173
346	OR TIME GREATER THAN OR EQUAL	OR >=DT	---	---	---	173
350	AND BIT TEST	AND TST	---	---	---	70
350	LOAD BIT TEST	LD TST	---	---	---	68
350	OR BIT TEST	OR TST	---	---	---	72
351	AND BIT TEST	AND TSTN	---	---	---	70
351	LOAD BIT TEST	LD TSTN	---	---	---	68
351	OR BIT TEST	OR TSTN	---	---	---	72
360	UNSIGNED ONE-WORD RECORD SEARCH LESS THAN	RSRCH <	@RSRCH <	---	---	534
361	UNSIGNED 0NE-WORD RECORD SEARCH LESS THAN OR EQUALS	RSRCH <=	@RSRCH <=	---	---	534
362	UNSIGNED 0NE-WORD RECORD SEARCH EQUALS	RSRCH =	@RSRCH =	---	---	534
363	UNSIGNED 0NE-WORD RECORD SEARCH GREATER THAN	RSRCH >	@RSRCH >	---	---	534
364	UNSIGNED 0NE-WORD RECORD SEARCH GREATER THAN OR EQUALS	RSRCH >=	@RSRCH >=	---	---	534
370	UNSIGNED TWO-WORD RECORD SEARCH LESS THAN	RSRCH2 <	@RSRCH2 <	---	---	540

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
371	UNSIGNED TWO-WORD RECORD SEARCH LESS THAN OR EQUALS	RSRCH2 <=	@RSRCH2 <=	---	---	540
372	UNSIGNED TWO-WORD RECORD SEARCH EQUALS	RSRCH2 =	@RSRCH2 =	---	---	540
373	UNSIGNED TWO-WORD RECORD SEARCH GREATER THAN	RSRCH2 >	@RSRCH2 >	---	---	540
374	UNSIGNED TWO-WORD RECORD SEARCH GREATER THAN OR EQUALS	RSRCH2 >=	@RSRCH2 >=	---	---	540
380	UNSIGNED FOUR-WORD RECORD SEARCH LESS THAN	RSRCH4 <	@RSRCH4 <	---	---	543
381	UNSIGNED FOUR-WORD RECORD SEARCH LESS THAN OR EQUALS	RSRCH4 <=	@RSRCH4 <=	---	---	543
382	UNSIGNED FOUR-WORD RECORD SEARCH EQUALS	RSRCH4 =	@RSRCH4 =	---	---	543
383	UNSIGNED FOUR-WORD RECORD SEARCH GREATER THAN	RSRCH4 >	@RSRCH4 >	---	---	543
384	UNSIGNED FOUR-WORD RECORD SEARCH GREATER THAN OR EQUALS	RSRCH4 >=	@RSRCH4 >=	---	---	543
400	SIGNED BINARY ADD WITHOUT CARRY	+	@+	---	---	266
401	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	@+L	---	---	266
402	SIGNED BINARY ADD WITH CARRY	+C	@+C	---	---	268
403	DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	@+CL	---	---	268
404	BCD ADD WITHOUT CARRY	+B	@+B	---	---	270
405	DOUBLE BCD ADD WITHOUT CARRY	+BL	@+BL	---	---	270
406	BCD ADD WITH CARRY	+BC	@+BC	---	---	272
407	DOUBLE BCD ADD WITH CARRY	+BCL	@+BCL	---	---	272
410	SIGNED BINARY SUBTRACT WITHOUT CARRY	–	@–	---	---	274
411	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	@–L	---	---	274
412	SIGNED BINARY SUBTRACT WITH CARRY	–C	@–C	---	---	278
413	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	@–CL	---	---	278
414	BCD SUBTRACT WITHOUT CARRY	–B	@–B	---	---	281
415	DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	@–BL	---	---	281
416	BCD SUBTRACT WITH CARRY	–BC	@–BC	---	---	284

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
417	DOUBLE BCD SUBTRACT WITH CARRY	−BCL	@−BCL	---	---	284
420	SIGNED BINARY MULTIPLY	*	@*	---	---	286
421	DOUBLE SIGNED BINARY MULTIPLY	*L	@*L	---	---	286
422	UNSIGNED BINARY MULTIPLY	*U	@*U	---	---	288
423	DOUBLE UNSIGNED BINARY MULTIPLY	*UL	@*UL	---	---	288
424	BCD MULTIPLY	*B	@*B	---	---	290
425	DOUBLE BCD MULTIPLY	*BL	@*BL	---	---	290
430	SIGNED BINARY DIVIDE	/	@/	---	---	292
431	DOUBLE SIGNED BINARY DIVIDE	/L	@/L	---	---	292
432	UNSIGNED BINARY DIVIDE	/U	@/U	---	---	294
433	DOUBLE UNSIGNED BINARY DIVIDE	/UL	@/UL	---	---	294
434	BCD DIVIDE	/B	@/B	---	---	296
435	DOUBLE BCD DIVIDE	/BL	@/BL	---	---	296
448	FLOATING POINT TO ASCII	FSTR	@FSTR	---	---	421
449	ASCII TO FLOATING POINT	FVAL	@FVAL	---	---	426
450	FLOATING TO 16-BIT	FIX	@FIX	---	---	388
451	FLOATING TO 32-BIT	FIXL	@FIXL	---	---	388
452	16-BIT TO FLOATING	FLT	@FLT	---	---	390
453	32-BIT TO FLOATING	FLTL	@FLTL	---	---	390
454	FLOATING-POINT ADD	+F	@+F	---	---	392
455	FLOATING-POINT SUBTRACT	−F	@−F	---	---	392
456	FLOATING-POINT MULTIPLY	*F	@*F	---	---	392
457	FLOATING-POINT DIVIDE	/F	@/F	---	---	392
458	DEGREES TO RADIANS	RAD	@RAD	---	---	396
459	RADIANS-TO DEGREES	DEG	@DEG	---	---	398
460	SINE	SIN	@SIN	---	---	400
461	COSINE	COS	@COS	---	---	400
462	TANGENT	TAN	@TAN	---	---	400
463	ARC SINE	ASIN	@ASIN	---	---	407
464	ARC COSINE	ACOS	@ACOS	---	---	407
465	ARC TANGENT	ATAN	@ATAN	---	---	407
466	SQUARE ROOT	SQRT	@SQRT	---	---	410
467	EXPONENT	EXP	@EXP	---	---	412
468	LOGARITHM	LOG	@LOG	---	---	414
469	MOVE FLOATING-POINT (SINGLE)	MOVF	@MOVF	---	---	430
470	SIGNED BCD TO BINARY	BINS	@BINS	---	---	330
471	SIGNED BINARY TO BCD	BCDS	@BCDS	---	---	335
472	DOUBLE SIGNED BCD TO BINARY	BISL	@BISL	---	---	330
473	DOUBLE SIGNED BINARY TO BCD	BDSL	@BDSL	---	---	335

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
474	GRAY CODE CONVERSION	GRY	@GRY	---	---	339
475	HIGH-SPEED SINE	SINQ	@SINQ	---	---	403
476	HIGH-SPEED COSINE	COSQ	@COSQ	---	---	403
477	HIGH-SPEED TANGENT	TANQ	@TANQ	---	---	403
478	GRAY CODE TO BINARY CONVERT	GRAY_BIN	@GRAY_BIN	---	---	344
479	DOUBLE GRAY CODE TO BINARY CONVERT	GRAY_BINL	@GRAY_BINL	---	---	344
480	BINARY TO GRAY CODE CONVERT	BIN_GRAY	@BIN_GRAY	---	---	346
481	DOUBLE BINARY TO GRAY CODE CONVERT	BIN_GRAYL	@BIN_GRAYL	---	---	346
486	SCALING 2	SCL2	@SCL2	---	---	593
487	SCALING 3	SCL3	@SCL3	---	---	597
490	DELIVER COMMAND	CMND	@CMND	---	---	870
491	NETWORK SEND 2	SEND2	@SEND2	---	---	857
492	NETWORK RECEIVE 2	RECV2	@RECV2	---	---	866
493	DELIVER COMMAND 2	CMND2	@CMND2	---	---	878
498	DOUBLE MOVE	MOVL	@MOVL	---	---	199
499	DOUBLE MOVE NOT	MVNL	@MVNL	---	---	202
502	EXPANDED BLOCK COMPARE	BCMP2	@BCMP2	---	---	189
510	CONDITIONAL JUMP	CJP	---	---	---	115
511	CONDITIONAL JUMP	CJPN	---	---	---	115
512	FOR-NEXT LOOPS	FOR	---	---	---	121
513	FOR-NEXT LOOPS	NEXT	---	---	---	121
514	BREAK LOOP	BREAK	---	---	---	124
515	MULTIPLE JUMP	JMP0	---	---	---	118
516	MULTIPLE JUMP END	JME0	---	---	---	118
517	MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	---	---	---	103
518	MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	---	---	---	103
519	MULTI-INTERLOCK CLEAR	MILC	---	---	---	103
520	NOT	NOT	---	---	---	65
521	CONDITION ON	UP	---	---	---	66
522	CONDITION OFF	DOWN	---	---	---	66
530	MULTIPLE BIT SET	SETA	@SETA	---	---	88
531	MULTIPLE BIT RESET	RSTA	@RSTA	---	---	88
532	SINGLE BIT SET	SETB	@SETB	---	!SETB	90
533	SINGLE BIT RESET	RSTB	@RSTB	---	!RSTB	90
534	SINGLE BIT OUTPUT	OUTB	@OUTB	---	!OUTB	92
540	ONE-MS TIMER	TMHH	---	---	---	141
541	TENTH-MS TIMER	TIMU	---	---	---	144
542	LONG TIMER	TIML	---	---	---	153
543	MULTI-OUTPUT TIMER	MTIM	---	---	---	156
544	HUNDREDTH-MS TIMER	TMUH	---	---	---	147
545	RESET TIMER/COUNTER	CNR	@CNR	---	---	166
546	COUNTER	CNTX	---	---	---	160

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
547	RESET TIMER/ COUNTER	CNRX	@CNRX	---	---	166
548	REVERSIBLE COUNTER	CNTRX	---	---	---	163
549	TIMER RESET	TRSET	@TRSET	---	---	168
550	HUNDRED-MS TIMER	TIMX	---	---	---	133
551	TEN-MS TIMER	TIMHX	---	---	---	137
552	ONE-MS TIMER	TMHHX	---	---	---	141
553	LONG TIMER	TIMLX	---	---	---	153
554	MULTI-OUTPUT TIMER	MTIMX	---	---	---	156
555	ACCUMULATIVE TIMER	TTIMX	---	---	---	150
556	TENTH-MS TIMER	TIMUX	---	---	---	144
557	HUNDREDTH-MS TIMER	TMUHX	---	---	---	147
560	MOVE TO REGISTER	MOVR	@MOVR	---	---	220
561	MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	---	---	---	220
562	DOUBLE DATA EXCHANGE	XCGL	@XCGL	---	---	214
565	BLOCK TRANSFER	XFERC	@XFERC	---	---	1044
566	SINGLE WORD DISTRIBUTE	DISTC	@DISTC	---	---	1046
567	DATA COLLECT	COLLC	@COLLC	---	---	1049
568	MOVE BIT	MOVBC	@MOVBC	---	---	1052
570	DOUBLE SHIFT LEFT	ASLL	@ASLL	---	---	231
571	DOUBLE SHIFT RIGHT	ASRL	@ASRL	---	---	233
572	DOUBLE ROTATE LEFT	ROLL	@ROLL	---	---	235
573	DOUBLE ROTATE RIGHT	RORL	@RORL	---	---	239
574	ROTATE LEFT WITHOUT CARRY	RLNC	@RLNC	---	---	237
575	ROTATE RIGHT WITHOUT CARRY	RRNC	@RRNC	---	---	241
576	DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	@RLNL	---	---	237
577	DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	@RRNL	---	---	241
578	SHIFT N-BIT DATA LEFT	NSFL	@NSFL	---	---	245
579	SHIFT N-BIT DATA RIGHT	NSFR	@NSFR	---	---	245
580	SHIFT N-BITS LEFT	NASL	@NASL	---	---	248
581	SHIFT N-BITS RIGHT	NASR	@NASR	---	---	251
582	DOUBLE SHIFT N-BITS LEFT	NSLL	@NSLL	---	---	248
583	DOUBLE SHIFT N-BITS RIGHT	NSRL	@NSRL	---	---	251
590	INCREMENT BINARY	++	@++	---	---	254
591	DOUBLE INCREMENT BINARY	++L	@++L	---	---	254
592	DECREMENT BINARY	– –	@– –	---	---	257
593	DOUBLE DECREMENT BINARY	– –L	@– –L	---	---	257
594	INCREMENT BCD	++B	@++B	---	---	260
595	DOUBLE INCREMENT BCD	++BL	@++BL	---	---	260

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
596	DECREMENT BCD	– –B	@– –B	---	---	263
597	DOUBLE DECREMENT BCD	– –BL	@– –BL	---	---	263
600	16-BIT TO 32-BIT SIGNED BINARY	SIGN	@SIGN	---	---	305
601	FOUR-DIGIT NUMBER TO ASCII	STR4	@STR4	---	---	348
602	EIGHT-DIGIT NUMBER TO ASCII	STR8	@STR8	---	---	348
603	SIXTEEN-DIGIT NUMBER TO ASCII	STR16	@STR16	---	---	348
604	ASCII TO FOUR-DIGIT NUMBER	NUM4	@NUM4	---	---	351
605	ASCII TO EIGHT-DIGIT NUMBER	NUM8	@NUM8	---	---	351
606	ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	@NUM16	---	---	351
610	DOUBLE LOGICAL AND	ANDL	@ANDL	---	---	354
611	DOUBLE LOGICAL OR	ORWL	@ORWL	---	---	356
612	DOUBLE EXCLUSIVE OR	XORL	@XORL	---	---	358
613	DOUBLE EXCLUSIVE NOR	XNRL	@XNRL	---	---	360
614	DOUBLE COMPLEMENT	COML	@COML	---	---	362
620	BINARY ROOT	ROTB	@ROTB	---	---	364
621	BIT COUNTER	BCNTC	@BCNTC	---	---	1054
630	SET STACK	SSET	@SSET	---	---	470
631	DIMENSION RECORD TABLE	DIM	@DIM	---	---	493
632	PUSH ONTO STACK	PUSH	@PUSH	---	---	473
633	FIRST IN FIRST OUT	FIFO	@FIFO	---	---	475
634	LAST IN FIRST OUT	LIFO	@LIFO	---	---	475
635	SET RECORD LOCATION	SETR	@SETR	---	---	495
636	GET RECORD NUMBER	GETR	@GETR	---	---	497
637	SWAP BYTES	SWAP	@SWAP	---	---	502
638	STACK SIZE READ	SNUM	@SNUM	---	---	479
639	STACK DATA READ	SREAD	@SREAD	---	---	481
640	STACK DATA WRITE	SWRIT	@SWRIT	---	---	484
641	STACK DATA INSERT	SINS	@SINS	---	---	487
642	STACK DATA DELETE	SDEL	@SDEL	---	---	490
650	STRING LENGTH	LEN$	@LEN$	---	---	1021
652	GET STRING LEFT	LEFT$	@LEFT$	---	---	1014
653	GET STRING RIGHT	RGHT$	@RGHT$	---	---	1014
654	GET STRING MIDDLE	MID$	@MID$	---	---	1017
656	CONCATENATE STRING	+$	@+$	---	---	1012
657	INS$	INS$	@INS$	---	---	1031
658	DELETE STRING	DEL$	@DEL$	---	---	1025
660	FIND IN STRING	FIND$	@FIND$	---	---	1019
661	REPLACE IN STRING	RPLC$	@RPLC$	---	---	1023
664	MOVE STRING	MOV$	@MOV$	---	---	1011
665	EXCHANGE STRING	XCHG$	@XCHG$	---	---	1027
666	CLEAR STRING	CLR$	@CLR$	---	---	1029
670	AND STRING EQUALS	AND =$	---	---	---	1033
670	LOAD STRING EQUALS	LD =$	---	---	---	1033

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
670	OR STRING EQUALS	OR =$	---	---	---	1033
671	AND STRING NOT EQUAL	AND <>$	---	---	---	1033
671	LOAD STRING NOT EQUAL	LD <>$	---	---	---	1033
671	OR STRING NOT EQUAL	OR <>$	---	---	---	1033
672	AND STRING LESS THAN	AND <$	---	---	---	1033
672	LOAD STRING LESS THAN	LD <$	---	---	---	1033
672	OR STRING LESS THAN	OR <$	---	---	---	1033
673	AND STRING LESS THAN OR EQUAL	AND <=$	---	---	---	1033
673	LOAD STRING LESS THAN OR EQUAL	LD <=$	---	---	---	1033
673	OR STRING LESS THAN OR EQUALS	OR <=$	---	---	---	1033
674	AND STRING GREATER THAN	AND >$	---	---	---	1033
674	LOAD STRING GREATER THAN	LD >$	---	---	---	1033
674	OR STRING GREATER THAN	OR >$	---	---	---	1033
675	AND STRING GREATER THAN OR EQUALS	AND >=$	---	---	---	1033
675	LOAD STRING GREATER THAN OR EQUALS	LD >=$	---	---	---	1033
675	OR STRING GREATER THAN OR EQUALS	OR >=$	---	---	---	1033
680	LIMIT CONTROL	LMT	@LMT	---	---	575
681	DEAD BAND CONTROL	BAND	@BAND	---	---	577
682	DEAD ZONE CONTROL	ZONE	@ZONE	---	---	580
685	TIME-PROPORTIONAL OUTPUT	TPO	---	---	---	582
690	SET INTERRUPT MASK	MSKS	@MSKS	---	---	628
691	CLEAR INTERRUPT	CLI	@CLI	---	---	639
692	READ INTERRUPT MASK	MSKR	@MSKR	---	---	634
693	DISABLE INTERRUPTS	DI	@DI	---	---	643
694	ENABLE INTERRUPTS	EI	---	---	---	645
700	READ DATA FILE	FREAD	@FREAD	---	---	911
701	WRITE DATA FILE	FWRIT	@FWRIT	---	---	916
704	WRITE TEXT FILE	TWRIT	@TWRIT	---	---	922
720	EXPLICIT MESSAGE SEND	EXPLT	@EXPLT	---	---	882
721	EXPLICIT GET ATTRIBUTE	EGATR	@EGATR	---	---	888
722	EXPLICIT SET ATTRIBUTE	ESATR	@ESATR	---	---	893
723	EXPLICIT WORD READ	ECHRD	@ECHRD	---	---	897
724	EXPLICIT WORD WRITE	ECHWR	@ECHWR	---	---	900
730	CALENDAR ADD	CADD	@CADD	---	---	930
731	CALENDAR SUBTRACT	CSUB	@CSUB	---	---	930

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
735	CLOCK ADJUSTMENT	DATE	@DATE	---	---	939
750	GLOBAL SUBROUTINE CALL	GSBS	@GSBS	---	---	616
751	GLOBAL SUBROUTINE ENTRY	GSBN	---	---	---	622
752	GLOBAL SUBROUTINE RETURN	GRET	---	---	---	622
780	READ SET TIMER	TSR	@TSR	---	---	1062
781	SET STEP TIMER	TSW	@TSW	---	---	1062
784	STEP ACTIVATE	SA	@SA	---	---	1060
785	STEP DEACTIVATE	SE	@SE	---	---	1060
789	SFC ON	SFCON	---	---	---	1064
790	SFC OFF	SFCOFF	---	---	---	1064
791	SFC PAUSE WITH NO RESET	SFCPRN	---	---	---	1066
793	SFC PAUSE WITH RESET	SFCPR	---	---	---	1066
801	BLOCK PROGRAM END	BEND	---	---	---	986
802	IF NOT	IF NOT (operand)	---	---	---	992
802	IF	IF (input condition)	---	---	---	992
802	IF	IF (operand)	---	---	---	992
803	ELSE	ELSE	---	---	---	992
804	IF END	IEND	---	---	---	992
805	ONE CYCLE AND WAIT NOT	WAIT NOT (operand)	---	---	---	995
805	ONE CYCLE AND WAIT	WAIT (input condition)	---	---	---	995
805	ONE CYCLE AND WAIT	WAIT (operand)	---	---	---	995
806	CONDITIONAL BLOCK EXIT NOT	EXIT NOT (operand)	---	---	---	990
806	CONDITIONAL BLOCK EXIT	EXIT (input condition)	---	---	---	990
806	CONDITIONAL BLOCK EXIT	EXIT (operand)	---	---	---	990
809	LOOP	LOOP	---	---	---	1006
810	LOOP END NOT	LEND NOT (operand)	---	---	---	1006
810	LOOP END	LEND (input condition)	---	---	---	1006
810	LOOP END	LEND (operand)	---	---	---	1006
811	BLOCK PROGRAM PAUSE	BPPS	---	---	---	988
812	BLOCK PROGRAM RESTART	BPRS	---	---	---	988
813	HUNDRED-MS TIMER WAIT	TIMW	---	---	---	998
814	COUNTER WAIT	CNTW	---	---	---	1001
815	TEN-MS TIMER WAIT	TMHW	---	---	---	1003
816	HUNDRED-MS TIMER WAIT	TIMWX	---	---	---	998
817	TEN-MS TIMER WAIT	TMHWX	---	---	---	1003
818	COUNTER WAIT	CNTWX	---	---	---	1001
820	TASK ON	TKON	@TKON	---	---	1038
821	TASK OFF	TKOF	@TKOF	---	---	1038
840	EXPONENTIAL POWER	PWR	@PWR	---	---	416
841	DOUBLE FLOATING TO 16-BIT BINARY	FIXD	@FIXD	---	---	437

Function code	Instruction	Mnemonic	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
842	DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	@FIXLD	---	---	437
843	16-BIT BINARY TO DOUBLE FLOATING	DBL	@DBL	---	---	439
844	32-BIT BINARY TO DOUBLE FLOATING	DBLL	@DBLL	---	---	439
845	DOUBLE FLOATING-POINT ADD	+D	@+D	---	---	441
846	DOUBLE FLOATING-POINT SUBTRACT	−D	@−D	---	---	441
847	DOUBLE FLOATING-POINT MULTIPLY	∗D	@∗D	---	---	441
848	DOUBLE FLOATING-POINT DIVIDE	/D	@/D	---	---	441
849	DOUBLE DEGREES TO RADIANS	RADD	@RADD	---	---	444
850	DOUBLE RADIANS TO DEGREES	DEGD	@RADD	---	---	446
851	DOUBLE SINE	SIND	@SIND	---	---	448
852	DOUBLE COSINE	COSD	@COSD	---	---	448
853	DOUBLE TANGENT	TAND	@TAND	---	---	448
854	DOUBLE ARC SINE	ASIND	@ASIND	---	---	451
855	DOUBLE ARC COSINE	ACOSD	@ACOSD	---	---	451
856	DOUBLE ARC TANGENT	ATAND	@ATAND	---	---	451
857	DOUBLE SQUARE ROOT	SQRTD	@SQRTD	---	---	454
858	DOUBLE EXPONENT	EXPD	@EXPD	---	---	456
859	DOUBLE LOGARITHM	LOGD	@LOGD	---	---	458
860	DOUBLE EXPONENTIAL POWER	PWRD	@PWRD	---	---	460
880	MODE CONTROL	INI	@INI	---	---	647
881	HIGH-SPEED COUNTER PV READ	PRV	@PRV	---	---	653
882	COMPARISON TABLE LOAD	CTBL	@CTBL	---	---	662
883	COUNTER FREQUENCY CONVERT	PRV2	@PRV2	---	---	659
885	SPEED OUTPUT	SPED	@SPED	---	---	669
886	SET PULSES	PULS	@PULS	---	---	673
887	PULSE OUTPUT	PLS2	@PLS2	---	---	675
888	ACCELERATION CONTROL	ACC	@ACC	---	---	684
889	ORIGIN SEARCH	ORG	@ORG	---	---	692
891	PULSE WITH VARIABLE DUTY FACTOR	PWM	@PWM	---	---	695
892	INTERRUPT FEEDING	IFEED	@IFEED	---	---	698

Appendix B

Alphabetical List of Instructions by Mnemonic

A

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ACC	ACCELERATION CONTROL	888	@ACC	---	---	684
ACOS	ARC COSINE	464	@ACOS	---	---	407
ACOSD	DOUBLE ARC COSINE	855	@ACOSD	---	---	451
AIDC	ANALOG INPUT DIRECT CONVERSION	216	@AIDC	---	---	744
AND	AND	---	@AND	%AND	!AND	58
AND <	AND LESS THAN	310	---	---	---	169
AND <$	AND STRING LESS THAN	672	---	---	---	1033
AND <>	AND NOT EQUAL	305	---	---	---	169
AND <>$	AND STRING NOT EQUAL	671	---	---	---	1033
AND <>D	AND DOUBLE FLOATING NOT EQUAL	336	---	---	---	462
AND <> DT	AND TIME NOT EQUAL	342	---	---	---	173
AND <>F	AND FLOATING NOT EQUAL	330	---	---	---	418
AND <>L	AND DOUBLE NOT EQUAL	306	---	---	---	169
AND <>S	AND SIGNED NOT EQUAL	307	---	---	---	169
AND <>SL	AND DOUBLE SIGNED NOT EQUAL	308	---	---	---	169
AND <D	AND DOUBLE FLOATING LESS THAN	337	---	---	---	462
AND <DT	AND TIME LESS THAN	343	---	---	---	173
AND <F	AND FLOATING LESS THAN	331	---	---	---	418
AND <L	AND DOUBLE LESS THAN	311	---	---	---	169
AND <S	AND SIGNED LESS THAN	312	---	---	---	169
AND <SL	AND DOUBLE SIGNED LESS THAN	313	---	---	---	169
AND =	AND EQUAL	300	---	---	---	169
AND =$	AND STRING EQUALS	670	---	---	---	1033
AND =D	AND DOUBLE FLOATING EQUAL	335	---	---	---	462
AND =DT	AND TIME EQUAL	341	---	---	---	173
AND =F	AND FLOATING EQUAL	329	---	---	---	418
AND =L	AND DOUBLE EQUAL	301	---	---	---	169
AND =S	AND SIGNED EQUAL	302	---	---	---	169
AND =SL	AND DOUBLE SIGNED EQUAL	303	---	---	---	169
AND >	AND GREATER THAN	320	---	---	---	169
AND >$	AND STRING GREATER THAN	674	---	---	---	1033

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
AND >D	AND DOUBLE FLOATING GREATER THAN	339	---	---	---	462
AND >DT	AND TIME GREATER THAN	345	---	---	---	173
AND >F	AND FLOATING GREATER THAN	333	---	---	---	418
AND >L	AND DOUBLE GREATER THAN	321	---	---	---	169
AND >S	AND SIGNED GREATER THAN	322	---	---	---	169
AND >SL	AND DOUBLE SIGNED GREATER THAN	323	---	---	---	169
AND LD	AND LOAD	---	---	---	---	62
AND NOT	AND NOT	---	---	---	!AND NOT	58
AND TST	AND BIT TEST	350	---	---	---	70
AND TSTN	AND BIT TEST	351	---	---	---	70
AND <=	AND LESS THAN OR EQUAL	315	---	---	---	169
AND <=$	AND STRING LESS THAN OR EQUAL	673	---	---	---	1033
AND <=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	---	462
AND <=DT	AND TIME LESS THAN OR EQUAL	344	---	---	---	173
AND <=F	AND FLOATING LESS THAN OR EQUAL	332	---	---	---	418
AND <=L	AND DOUBLE LESS THAN OR EQUAL	316	---	---	---	169
AND <=S	AND SIGNED LESS THAN OR EQUAL	317	---	---	---	169
AND <=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	---	169
AND >=	AND GREATER THAN OR EQUAL	325	---	---	---	169
AND >=$	AND STRING GREATER THAN OR EQUALS	675	---	---	---	1033
AND >=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	---	462
AND >=DT	AND TIME GREATER THAN OR EQUAL	346	---	---	---	173
AND >=F	AND FLOATING GREATER THAN OR EQUAL	334	---	---	---	418
AND >=L	AND DOUBLE GREATER THAN OR EQUAL	326	---	---	---	169
AND >=S	AND SIGNED GREATER THAN OR EQUAL	327	---	---	---	169
AND >=SL	AND DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	---	169
ANDL	DOUBLE LOGICAL AND	610	@ANDL	---	---	354
ANDW	LOGICAL AND	034	@ANDW	---	---	354
AODC	ANALOG OUTPUT DIRECT CONVERSION	217	@AODC	---	---	747
APR	ARITHMETIC PROCESS	069	@APR	---	---	369
ASC	ASCII CONVERT	086	@ASC	---	---	317

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ASFT	ASYNCHRONOUS SHIFT REGISTER	017	@ASFT	---	---	227
ASIN	ARC SINE	463	@ASIN	---	---	407
ASIND	DOUBLE ARC SINE	854	@ASIND	---	---	451
ASL	ARITHMETIC SHIFT LEFT	025	@ASL	---	---	231
ASLL	DOUBLE SHIFT LEFT	570	@ASLL	---	---	231
ASR	ARITHMETIC SHIFT RIGHT	026	@ASR	---	---	233
ASRL	DOUBLE SHIFT RIGHT	571	@ASRL	---	---	233
ATAN	ARC TANGENT	465	@ATAN	---	---	407
ATAND	DOUBLE ARC TANGENT	856	@ATAND	---	---	451
AVG	AVERAGE	195	---	---	---	600

B

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
BAND	DEAD BAND CONTROL	681	@BAND	---	---	577
BCD	BINARY TO BCD	024	@BCD	---	---	300
BCDL	DOUBLE BINARY TO BCD	059	@BCDL	---	---	300
BCDS	SIGNED BINARY TO BCD	471	@BCDS	---	---	335
BCMP	UNSIGNED BLOCK COMPARE	068	@BCMP	---	---	187
BCMP2	EXPANDED BLOCK COMPARE	502	@BCMP2	---	---	189
BCNT	BIT COUNTER	067	@BCNT	---	---	381
BCNTC	BIT COUNTER	621	@BCNTC	---	---	1054
BDSL	DOUBLE SIGNED BINARY TO BCD	473	@BDSL	---	---	335
BEND	BLOCK PROGRAM END	801	---	---	---	986
BIN	BCD TO BINARY	023	@BIN	---	---	298
BIN_GRAY	BINARY TO GRAY CODE CONVERT	480	@BIN_GRAY	---	---	346
BIN_GRAYL	DOUBLE BINARY TO GRAY CODE CONVERT	481	@BIN_GRAYL	---	---	346
BINL	DOUBLE BCD TO DOUBLE BINARY	058	@BINL	---	---	298
BINS	SIGNED BCD TO BINARY	470	@BINS	---	---	330
BISL	DOUBLE SIGNED BCD TO BINARY	472	@BISL	---	---	330
BPPS	BLOCK PROGRAM PAUSE	811	---	---	---	988
BPRG	BLOCK PROGRAM BEGIN	096	---	---	---	986
BPRS	BLOCK PROGRAM RESTART	812	---	---	---	988
BREAK	BREAK LOOP	514	---	---	---	124
BSET	BLOCK SET	071	@BSET	---	---	212

C

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
CADD	CALENDAR ADD	730	@CADD	---	---	930
CCL	LOAD CONDITION FLAGS	283	@CCL	---	---	970
CCS	SAVE CONDITION FLAGS	282	@CCS	---	---	970
CJP	CONDITIONAL JUMP	510	---	---	---	115
CJPN	CONDITIONAL JUMP	511	---	---	---	115
CLC	CLEAR CARRY	041	@CLC	---	---	965
CLI	CLEAR INTERRUPT	691	@CLI	---	---	639
CLR$	CLEAR STRING	666	@CLR$	---	---	1029
CMND	DELIVER COMMAND	490	@CMND	---	---	870
CMND2	DELIVER COMMAND 2	493	@CMND2	---	---	878
CMP	COMPARE	020	---	---	!CMP	177
CMPL	DOUBLE COMPARE	060	---	---	---	177
CNR	RESET TIMER/ COUNTER	545	@CNR	---	---	166
CNRX	RESET TIMER/ COUNTER	547	@CNRX	---	---	166
CNT	COUNTER	---	---	---	---	160
CNTX	COUNTER	546	---	---	---	160
CNTR	REVERSIBLE COUNTER	012	---	---	---	163
CNTRX	REVERSIBLE COUNTER	548	---	---	---	163
CNTW	COUNTER WAIT	814	---	---	---	1001
CNTWX	COUNTER WAIT	818	---	---	---	1001
COLL	DATA COLLECT	081	@COLL	---	---	218
COLLC	DATA COLLECT	567	@COLLC	---	---	1049
COLM	LINE TO COLUMN	064	@COLM	---	---	328
COM	COMPLEMENT	029	---	---	---	362
COML	DOUBLE COMPLEMENT	614	@COML	---	---	362
COS	COSINE	461	@COS	---	---	400
COSD	DOUBLE COSINE	852	@COSD	---	---	448
COSQ	HIGH-SPEED COSINE	476	@COSQ	---	---	403
CPS	SIGNED BINARY COMPARE	114	---	---	!CPS	180
CPSL	DOUBLE SIGNED BINARY COMPARE	115	---	---	---	180
CSUB	CALENDAR SUBTRACT	731	@CSUB	---	---	930
CTBL	COMPARISON TABLE LOAD	882	@CTBL	---	---	662

D

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
DATE	CLOCK ADJUSTMENT	735	@DATE	---	---	939
DBL	16-BIT BINARY TO DOUBLE FLOATING	843	@DBL	---	---	439
DBLL	32-BIT BINARY TO DOUBLE FLOATING	844	@DBLL	---	---	439
DEG	RADIANS-TO DEGREES	459	@DEG	---	---	398
DEGD	DOUBLE RADIANS TO DEGREES	850	@RADD	---	---	446
DEL$	DELETE STRING	658	@DEL$	---	---	1025

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
DI	DISABLE INTERRUPTS	693	@DI	---	---	643
DIFD	DIFFERENTIATE DOWN	014	---	---	!DIFD	84
DIFU	DIFFERENTIATE UP	013	---	---	!DIFU	82
DIM	DIMENSION RECORD TABLE	631	@DIM	---	---	493
DIST	SINGLE WORD DISTRIBUTE	080	@DIST	---	---	216
DISTC	SINGLE WORD DISTRIBUTE	566	@DISTC	---	---	1046
DLNK	CPU BUS UNIT I/O REFRESH	226	@DLNK	---	---	718
DMPX	DATA ENCODER	077	@DMPX	---	---	312
DOWN	CONDITION OFF	522	---	---	---	66
DRXDU	DIRECT RECEIVE VIA SERIAL COMMUNICATIONS UNIT	261	@DRXDU	---	---	809
DSW	DIGITAL SWITCH INPUT	210	---	---	---	725
DTXDU	DIRECT TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	262	@DTXDU	---	---	803

E

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ECHRD	EXPLICIT WORD READ	723	@ECHRD	---	---	897
ECHWR	EXPLICIT WORD WRITE	724	@ECHWR	---	---	900
EGATR	EXPLICIT GET ATTRIBUTE	721	@EGATR	---	---	888
EI	ENABLE INTERRUPTS	694	---	---	---	645
ELSE	ELSE	803	---	---	---	992
EMBC	SELECT EM BANK	281	@EMBC	---	---	966
END	END	001	---	---	---	97
ESATR	EXPLICIT SET ATTRIBUTE	722	@ESATR	---	---	893
EXIT NOT (operand)	CONDITIONAL BLOCK EXIT NOT	806	---	---	---	990
EXIT (input condition)	CONDITIONAL BLOCK EXIT	806	---	---	---	990
EXIT (operand)	CONDITIONAL BLOCK EXIT	806	---	---	---	990
EXP	EXPONENT	467	@EXP	---	---	412
EXPD	DOUBLE EXPONENT	858	@EXPD	---	---	456
EXPLT	EXPLICIT MESSAGE SEND	720	@EXPLT	---	---	882

F

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
FAL	FAILURE ALARM	006	@FAL	---	---	944
FALS	SEVERE FAILURE ALARM	007	---	---	---	951
FCS	FRAME CHECKSUM	180	@FCS	---	---	524
FDIV	FLOATING POINT DIVIDE	079	@FDIV	---	---	378
FIFO	FIRST IN FIRST OUT	633	@FIFO	---	---	475
FIND$	FIND IN STRING	660	@FIND$	---	---	1019

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
FIORF	SPECIAL I/O UNIT I/O REFRESH	225	@FIORF	---	---	715
FIX	FLOATING TO 16-BIT	450	@FIX	---	---	388
FIXD	DOUBLE FLOATING TO 16-BIT BINARY	841	@FIXD	---	---	437
FIXL	FLOATING TO 32-BIT	451	@FIXL	---	---	388
FIXLD	DOUBLE FLOATING TO 32-BIT BINARY	842	@FIXLD	---	---	437
FLT	16-BIT TO FLOATING	452	@FLT	---	---	390
FLTL	32-BIT TO FLOATING	453	@FLTL	---	---	390
FOR	FOR-NEXT LOOPS	512	---	---	---	121
FPD	FAILURE POINT DETECTION	269	---	---	---	957
FREAD	READ DATA FILE	700	@FREAD	---	---	911
FRMCV	CONVERT ADDRESS FROM CV	284	@FRMCV	---	---	973
FSTR	FLOATING POINT TO ASCII	448	@FSTR	---	---	421
FWRIT	WRITE DATA FILE	701	@FWRIT	---	---	916
FVAL	ASCII TO FLOATING POINT	449	@FVAL	---	---	426

G

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
GETID	GET VARIABLE ID	286	@GETID	---	---	1056
GETR	GET RECORD NUMBER	636	@GETR	---	---	497
GRAY_BIN	GRAY CODE TO BINARY CONVERT	478	@GRAY_BIN	---	---	344
GRAY_BINL	DOUBLE GRAY CODE TO BINARY CONVERT	479	@GRAY_BINL	---	---	344
GRET	GLOBAL SUBROUTINE RETURN	752	---	---	---	622
GRY	GRAY CODE CONVERSION	474	@GRY	---	---	339
GSBN	GLOBAL SUBROUTINE ENTRY	751	---	---	---	622
GSBS	GLOBAL SUBROUTINE CALL	750	@GSBS	---	---	616

H

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
HEX	ASCII TO HEX	162	@HEX	---	---	321
HKY	HEXADECIMAL KEY INPUT	212	---	---	---	732
HMS	SECONDS TO HOURS	066	@HMS	---	---	937

I

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
IEND	IF END	804	---	---	---	992
IFEED	INTERRUPT FEEDING	892	@IFEED	---	---	698
IF NOT (operand)	IF NOT	802	---	---	---	992
IF (input condition)	IF	802	---	---	---	992
IF (operand)	IF	802	---	---	---	992
IL	INTERLOCK	002	---	---	---	99

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ILC	INTERLOCK CLEAR	003	---	---	---	99
INI	MODE CONTROL	880	@INI	---	---	647
INS$	INS$	657	@INS$	---	---	1031
IORD	INTELLIGENT I/O READ	222	@IORD	---	---	757
IORF	I/O REFRESH	097	@IORF	---	---	712
IORS	ENABLE PERIPHERAL SERVICING	288	---	---	---	980
IOSP	DISABLE PERIPHERAL SERVICING	287	@IOSP	---	---	980
IOWR	INTELLIGENT I/O WRITE	223	@IOWR	---	---	760

J

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
JME	JUMP END	005	---	---	---	112
JME0	MULTIPLE JUMP END	516	---	---	---	118
JMP	JUMP	004	---	---	---	112
JMP0	MULTIPLE JUMP	515	---	---	---	118

K

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
KEEP	KEEP	011	---	---	!KEEP	78

L

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
LD	LOAD	---	@LD	%LD	!LD	56
LD <	LOAD LESS THAN	310	---	---	---	169
LD <$	LOAD STRING LESS THAN	672	---	---	---	1033
LD <D	LOAD DOUBLE FLOATING LESS THAN	337	---	---	---	462
LD <DT	LOAD TIME LESS THAN	343	---	---	---	173
LD <F	LOAD FLOATING LESS THAN	331	---	---	---	418
LD <>	LOAD NOT EQUAL	305	---	---	---	169
LD <>$	LOAD STRING NOT EQUAL	671	---	---	---	1033
LD <>D	LOAD DOUBLE FLOATING NOT EQUAL	336	---	---	---	462
LD <>DT	LOAD TIME NOT EQUAL	342	---	---	---	173
LD <>F	LOAD FLOATING NOT EQUAL	330	---	---	---	418
LD <>L	LOAD DOUBLE NOT EQUAL	306	---	---	---	169
LD <>S	LOAD SIGNED NOT EQUAL	307	---	---	---	169
LD <>SL	LOAD DOUBLE SIGNED NOT EQUAL	308	---	---	---	169
LD <L	LOAD DOUBLE LESS THAN	311	---	---	---	169
LD <S	LOAD SIGNED LESS THAN	312	---	---	---	169

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
LD <SL	LOAD DOUBLE SIGNED LESS THAN	313	---	---	---	169
LD =	LOAD EQUAL	300	---	---	---	169
LD =$	LOAD STRING EQUALS	670	---	---	---	1033
LD =D	LOAD DOUBLE FLOATING EQUAL	335	---	---	---	462
LD =DT	LOAD TIME EQUAL	341	---	---	---	173
LD =F	LOAD FLOATING EQUAL	329	---	---	---	418
LD =L	LOAD DOUBLE EQUAL	301	---	---	---	169
LD =S	LOAD SIGNED EQUAL	302	---	---	---	169
LD =SL	LOAD DOUBLE SIGNED EQUAL	303	---	---	---	169
LD >	LOAD GREATER THAN	320	---	---	---	169
LD >$	LOAD STRING GREATER THAN	674	---	---	---	1033
LD >D	LOAD DOUBLE FLOATING GREATER THAN	339	---	---	---	462
LD >DT	LOAD TIME GREATER THAN	345	---	---	---	173
LD >F	LOAD FLOATING GREATER THAN	333	---	---	---	418
LD >L	LOAD DOUBLE GREATER THAN	321	---	---	---	169
LD >S	LOAD SIGNED GREATER THAN	322	---	---	---	169
LD >SL	LOAD DOUBLE SIGNED GREATER THAN	323	---	---	---	169
LD NOT	LOAD NOT	---	---	---	!LD NOT	56
LD TST	LOAD BIT TEST	350	---	---	---	68
LD TSTN	LOAD BIT TEST	351	---	---	---	68
LD <=	LOAD LESS THAN OR EQUAL	315	---	---	---	169
LD <=$	LOAD STRING LESS THAN OR EQUAL	673	---	---	---	1033
LD <=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	---	462
LD <=DT	LOAD TIME LESS THAN OR EQUAL	344	---	---	---	173
LD <=F	LOAD FLOATING LESS THAN OR EQUAL	332	---	---	---	418
LD <=L	LOAD DOUBLE LESS THAN OR EQUAL	316	---	---	---	169
LD <=S	LOAD SIGNED LESS THAN OR EQUAL	317	---	---	---	169
LD <=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	---	169
LD >=	LOAD GREATER THAN OR EQUAL	325	---	---	---	169
LD >=$	LOAD STRING GREATER THAN OR EQUALS	675	---	---	---	1033
LD >=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	---	462
LD >=DT	LOAD TIME GREATER THAN OR EQUAL	346	---	---	---	173

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
LD >=F	LOAD FLOATING GREATER THAN OR EQUAL	334	---	---	---	418
LD >=L	LOAD DOUBLE GREATER THAN OR EQUAL	326	---	---	---	169
LD >=S	LOAD SIGNED GREATER THAN OR EQUAL	327	---	---	---	169
LD >=SL	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	---	169
LEFT$	GET STRING LEFT	652	@LEFT$	---	---	1014
LEN$	STRING LENGTH	650	@LEN$	---	---	1021
LEND NOT (operand)	LOOP END NOT	810	---	---	---	1006
LEND (input condition)	LOOP END	810	---	---	---	1006
LEND (operand)	LOOP END	810	---	---	---	1006
LIFO	LAST IN FIRST OUT	634	@LIFO	---	---	475
LINE	COLUMN TO LINE	063	@LINE	---	---	326
LMT	LIMIT CONTROL	680	@LMT	---	---	575
LOG	LOGARITHM	468	@LOG	---	---	414
LOGD	DOUBLE LOGARITHM	859	@LOGD	---	---	458
LOOP	LOOP	809	---	---	---	1006

M

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
MAX	FIND MAXIMUM	182	@MAX	---	---	504
MAXD	FIND DOUBLE MAXIMUM FLOATING	178	@MAXD	---	---	513
MAXF	FIND MAXIMUM FLOATING	176	@MAXF	---	---	511
MAXL	DOUBLE FIND MAXIMUM	174	@MAXL	---	---	508
MCMP	MULTIPLE COMPARE	019	@MCMP	---	---	183
MCRO	MACRO	099	@MCRO	---	---	610
MID$	GET STRING MIDDLE	654	@MID$	---	---	1017
MILC	MULTI-INTERLOCK CLEAR	519	---	---	---	103
MILH	MULTI-INTERLOCK DIFFERENTIATION HOLD	517	---	---	---	103
MILR	MULTI-INTERLOCK DIFFERENTIATION RELEASE	518	---	---	---	103
MIN	FIND MINIMUM	183	@MIN	---	---	504
MIND	FIND DOUBLE MINIMUM FLOATING	179	@MIND	---	---	519
MINF	FIND MINIMUM FLOATING	177	@MINF	---	---	517
MINL	DOUBLE FIND MINIMUM	175	@MINL	---	---	515
MLPX	DATA DECODER	076	@MLPX	---	---	307
MOV	MOVE	021	@MOV	---	!MOV	199
MOV$	MOVE STRING	664	@MOV$	---	---	1011
MOVB	MOVE BIT	082	@MOVB	---	---	204
MOVBC	MOVE BIT	568	@MOVBC	---	---	1052
MOVD	MOVE DIGIT	083	@MOVD	---	---	206

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
MOVF	MOVE FLOATING-POINT (SINGLE)	469	@MOVF	---	---	430
MOVL	DOUBLE MOVE	498	@MOVL	---	---	199
MOVR	MOVE TO REGISTER	560	@MOVR	---	---	220
MOVRW	MOVE TIMER/ COUNTER PV TO REGISTER	561	---	---	---	220
MSG	DISPLAY MESSAGE	046	@MSG	---	---	927
MSKR	READ INTERRUPT MASK	692	@MSKR	---	---	634
MSKS	SET INTERRUPT MASK	690	@MSKS	---	---	628
MTIM	MULTI-OUTPUT TIMER	543	---	---	---	156
MTIMX	MULTI-OUTPUT TIMER	554	---	---	---	156
MTR	MATRIX INPUT	213	---	---	---	736
MVN	MOVE NOT	022	@MVN	---	---	202
MVNL	DOUBLE MOVE NOT	499	@MVNL	---	---	202

N

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
NASL	SHIFT N-BITS LEFT	580	@NASL	---	---	248
NASR	SHIFT N-BITS RIGHT	581	@NASR	---	---	251
NCDMV	PCU HIGH-SPEED POSITIONING	218	@NCDMV	---	---	750
NCDTR	PCU POSITIONING TRIGGER	219	@NCDTR	---	---	754
NEG	2'S COMPLEMENT	160	@NEG	---	---	303
NEGL	DOUBLE 2'S COMPLEMENT	161	@NEGL	---	---	303
NEXT	FOR-NEXT LOOPS	513	---	---	---	121
NOP	NO OPERATION	000	---	---	---	98
NOT	NOT	520	---	---	---	65
NSFL	SHIFT N-BIT DATA LEFT	578	@NSFL	---	---	245
NSFR	SHIFT N-BIT DATA RIGHT	579	@NSFR	---	---	245
NSLL	DOUBLE SHIFT N-BITS LEFT	582	@NSLL	---	---	248
NSRL	DOUBLE SHIFT N-BITS RIGHT	583	@NSRL	---	---	251
NUM4	ASCII TO FOUR-DIGIT NUMBER	604	@NUM4	---	---	351
NUM8	ASCII TO EIGHT-DIGIT NUMBER	605	@NUM8	---	---	351
NUM16	ASCII TO SIXTEEN-DIGIT NUMBER	606	@NUM16	---	---	351

O

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
OR	OR	---	@OR	%OR	!OR	60
OR <	OR LESS THAN	310	---	---	---	169
OR <$	OR STRING LESS THAN	672	---	---	---	1033
OR <>	OR NOT EQUAL	305	---	---	---	169
OR <>$	OR STRING NOT EQUAL	671	---	---	---	1033

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
OR <>D	OR DOUBLE FLOATING NOT EQUAL	336	---	---	---	462
OR <>DT	OR TIME NOT EQUAL	342	---	---	---	173
OR <>F	OR FLOATING NOT EQUAL	330	---	---	---	418
OR <>L	OR DOUBLE NOT EQUAL	306	---	---	---	169
OR <>S	OR SIGNED NOT EQUAL	307	---	---	---	169
OR <>SL	OR DOUBLE SIGNED NOT EQUAL	308	---	---	---	169
OR <D	OR DOUBLE FLOATING LESS THAN	337	---	---	---	462
OR <DT	OR TIME LESS THAN	343	---	---	---	173
OR <F	OR FLOATING LESS THAN	331	---	---	---	418
OR <L	OR DOUBLE LESS THAN	311	---	---	---	169
OR <S	OR SIGNED LESS THAN	312	---	---	---	169
OR <SL	OR DOUBLE SIGNED LESS THAN	313	---	---	---	169
OR =	OR EQUAL	300	---	---	---	169
OR =$	OR STRING EQUALS	670	---	---	---	1033
OR =D	OR DOUBLE FLOATING EQUAL	335	---	---	---	462
OR =DT	OR TIME EQUAL	341	---	---	---	173
OR =F	OR FLOATING EQUAL	329	---	---	---	418
OR =L	OR DOUBLE EQUAL	301	---	---	---	169
OR =S	OR SIGNED EQUAL	302	---	---	---	169
OR =SL	OR DOUBLE SIGNED EQUAL	303	---	---	---	169
OR >	OR GREATER THAN	320	---	---	---	169
OR >$	OR STRING GREATER THAN	674	---	---	---	1033
OR >D	OR DOUBLE FLOATING GREATER THAN	339	---	---	---	462
OR >DT	OR TIME GREATER THAN	345	---	---	---	173
OR >F	OR FLOATING GREATER THAN	333	---	---	---	418
OR >L	OR DOUBLE GREATER THAN	321	---	---	---	169
OR >S	OR SIGNED GREATER THAN	322	---	---	---	169
OR >SL	OR DOUBLE SIGNED GREATER THAN	323	---	---	---	169
OR LD	OR LOAD	---	---	---	---	62
OR NOT	OR NOT	---	---	---	!OR NOT	60
OR TST	OR BIT TEST	350	---	---	---	72
OR TSTN	OR BIT TEST	351	---	---	---	72
OR <=	OR LESS THAN OR EQUAL	315	---	---	---	169
OR <=$	OR STRING LESS THAN OR EQUALS	673	---	---	---	1033
OR <=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	---	462
OR <=DT	OR TIME LESS THAN OR EQUAL	344	---	---	---	173
OR <=F	OR FLOATING LESS THAN OR EQUAL	332	---	---	---	418
OR <=L	OR DOUBLE LESS THAN OR EQUAL	316	---	---	---	169

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
OR <=S	OR SIGNED LESS THAN OR EQUAL	317	---	---	---	169
OR <=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	---	169
OR >=	OR GREATER THAN OR EQUAL	325	---	---	---	169
OR >=$	OR STRING GREATER THAN OR EQUALS	675	---	---	---	1033
OR >=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	---	462
OR >=DT	OR TIME GREATER THAN OR EQUAL	346	---	---	---	173
OR >=F	OR FLOATING GREATER THAN OR EQUAL	334	---	---	---	418
OR >=L	OR DOUBLE GREATER THAN OR EQUAL	326	---	---	---	169
OR >=S	OR SIGNED GREATER THAN OR EQUAL	327	---	---	---	169
OR >=SL	OR DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	---	169
ORG	ORIGIN SEARCH	889	@ORG	---	---	692
ORW	LOGICAL OR	035	@ORW	---	---	356
ORWL	DOUBLE LOGICAL OR	611	@ORWL	---	---	356
OUT	OUTPUT	---	---	---	!OUT	74
OUTB	SINGLE BIT OUTPUT	534	@OUTB	---	!OUTB	92
OUT NOT	OUTPUT NOT	---	---	---	!OUT NOT	74

P

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
PID	PID CONTROL	190	---	---	---	556
PIDAT	PID CONTROL WITH AUTOTUNING	191	---	---	---	567
PMCR	PROTOCOL MACRO	260	@PMCR	---	---	766
PMCR2	PROTOCOL MACRO 2	264	@PMCR2	---	---	821
PRV	HIGH-SPEED COUNTER PV READ	881	@PRV	---	---	653
PRV2	COUNTER FREQUENCY CONVERT	883	@PRV2	---	---	659
PULS	SET PULSES	886	@PULS	---	---	673
PLS2	PULSE OUTPUT	887	@PLS2	---	---	675
PUSH	PUSH ONTO STACK	632	@PUSH	---	---	473
PWM	PULSE WITH VARIABLE DUTY FACTOR	891	@PWM	---	---	695
PWR	EXPONENTIAL POWER	840	@PWR	---	---	416
PWRD	DOUBLE EXPONENTIAL POWER	860	@PWRD	---	---	460

R

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
RAD	DEGREES TO RADIANS	458	@RAD	---	---	396
RADD	DOUBLE DEGREES TO RADIANS	849	@RADD	---	---	444

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
RECV	NETWORK RECEIVE	098	@RECV	---	---	861
RECV2	NETWORK RECEIVE 2	492	@RECV2	---	---	866
RET	SUBROUTINE RETURN	093	---	---	---	613
RGHT$	GET STRING RIGHT	653	@RGHT$	---	---	1014
RLNC	ROTATE LEFT WITHOUT CARRY	574	@RLNC	---	---	237
RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	576	@RLNL	---	---	237
ROL	ROTATE LEFT	027	@ROL	---	---	235
ROLL	DOUBLE ROTATE LEFT	572	@ROLL	---	---	235
ROOT	BCD SQUARE ROOT	072	@ROOT	---	---	366
ROR	ROTATE RIGHT	028	@ROR	---	---	239
RORL	DOUBLE ROTATE RIGHT	573	@RORL	---	---	239
ROTB	BINARY ROOT	620	@ROTB	---	---	364
RPLC$	REPLACE IN STRING	661	@RPLC$	---	---	1023
RRNC	ROTATE RIGHT WITHOUT CARRY	575	@RRNC	---	---	241
RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	577	@RRNL	---	---	241
RSET	RESET	---	@RSET	%RSET	!RSET	86
RSORT	UNSIGNED ONE-WORD RECORD SORT	203	@RSORT	---	---	546
RSORT2	UNSIGNED TWO-WORD RECORD SORT	204	@RSORT2	---	---	550
RSORT4	UNSIGNED FOUR-WORD RECORD SORT	205	@RSORT4	---	---	553
RSRCH <	UNSIGNED ONE-WORD RECORD SEARCH LESS THAN	360	@RSRCH <	---	---	534
RSRCH <=	UNSIGNED ONE-WORD RECORD SEARCH LESS THAN OR EQUALS	361	@RSRCH <=	---	---	534
RSRCH =	UNSIGNED ONE-WORD RECORD SEARCH EQUALS	362	@RSRCH =	---	---	534
RSRCH >	UNSIGNED ONE-WORD RECORD SEARCH GREATER THAN	363	@RSRCH >	---	---	534
RSRCH >=	UNSIGNED ONE-WORD RECORD SEARCH GREATER THAN OR EQUALS	364	@RSRCH >=	---	---	534
RSRCH2 <	UNSIGNED TWO-WORD RECORD SEARCH LESS THAN	370	@RSRCH2 <	---	---	540
RSRCH2 <=	UNSIGNED TWO-WORD RECORD SEARCH LESS THAN OR EQUALS	371	@RSRCH2 <=	---	---	540
RSRCH2 =	UNSIGNED TWO-WORD RECORD SEARCH EQUALS	372	@RSRCH2 =	---	---	540
RSRCH2 >	UNSIGNED TWO-WORD RECORD SEARCH GREATER THAN	373	@RSRCH2 >	---	---	540

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
RSRCH2 >=	UNSIGNED TWO-WORD RECORD SEARCH GREATER THAN OR EQUALS	374	@RSRCH2 >=	---	---	540
RSRCH4 <	UNSIGNED FOUR-WORD RECORD SEARCH LESS THAN	380	@RSRCH4 <	---	---	543
RSRCH4 <=	UNSIGNED FOUR-WORD RECORD SEARCH LESS THAN OR EQUALS	381	@RSRCH4 <=	---	---	543
RSRCH4 =	UNSIGNED FOUR-WORD RECORD SEARCH EQUALS	382	@RSRCH4 =	---	---	543
RSRCH4 >	UNSIGNED FOUR-WORD RECORD SEARCH GREATER THAN	383	@RSRCH4 >	---	---	543
RSRCH4 >=	UNSIGNED FOUR-WORD RECORD SEARCH GREATER THAN OR EQUALS	384	@RSRCH4 >=	---	---	543
RSTA	MULTIPLE BIT RESET	531	@RSTA	---	---	88
RSTB	SINGLE BIT RESET	533	@RSTB	---	!RSTB	90
RXD	RECEIVE	235	@RXD	---	---	779
RXDU	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	255	@RXDU	---	---	796

S

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
SA	STEP ACTIVATE	784	@SA	---	---	1060
SBN	SUBROUTINE ENTRY	092	---	---	---	613
SBS	SUBROUTINE CALL	091	@SBS	---	---	604
SCL	SCALING	194	@SCL	---	---	589
SCL2	SCALING 2	486	@SCL2	---	---	593
SCL3	SCALING 3	487	@SCL3	---	---	597
SDEC	7-SEGMENT DECODER	078	@SDEC	---	---	722
SDEL	STACK DATA DELETE	642	@SDEL	---	---	490
SE	STEP DEACTIVATE	785	@SE	---	---	1060
SEC	HOURS TO SECONDS	065	@SEC	---	---	935
SEND	NETWORK SEND	090	@SEND	---	---	851
SEND2	NETWORK SEND 2	491	@SEND2	---	---	857
SET	SET	---	@SET	%SET	!SET	86
SETA	MULTIPLE BIT SET	530	@SETA	---	---	88
SETB	SINGLE BIT SET	532	@SETB	---	!SETB	90
SETR	SET RECORD LOCATION	635	@SETR	---	---	495
SFCON	SFC ON	789	---	---	---	1064
SFCOFF	SFC OFF	790	---	---	---	1064
SFCPRN	SFC PAUSE WITH NO RESET	791	---	---	---	1066
SFCPR	SFC PAUSE WITH RESET	793	---	---	---	1066
SFT	SHIFT REGISTER	010	---	---	---	223
SFTR	REVERSIBLE SHIFT REGISTER	084	@SFTR	---	---	225
SIGN	16-BIT TO 32-BIT SIGNED BINARY	600	@SIGN	---	---	305
SIN	SINE	460	@SIN	---	---	400
SIND	DOUBLE SINE	851	@SIND	---	---	448

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
SINQ	HIGH-SPEED SINE	475	@SINQ	---	---	403
SINS	STACK DATA INSERT	641	@SINS	---	---	487
SLD	ONE DIGIT SHIFT LEFT	074	@SLD	---	---	243
SNUM	STACK SIZE READ	638	@SNUM	---	---	479
SNXT	STEP START	009	---	---	---	702
SPED	SPEED OUTPUT	885	@SPED	---	---	669
SQRT	SQUARE ROOT	466	@SQRT	---	---	410
SQRTD	DOUBLE SQUARE ROOT	857	@SQRTD	---	---	454
SRCH	DATA SEARCH	181	@SRCH	---	---	499
SRD	ONE DIGIT SHIFT RIGHT	075	@SRD	---	---	243
SREAD	STACK DATA READ	639	@SREAD	---	---	481
SSET	SET STACK	630	@SSET	---	---	470
STC	SET CARRY	040	@STC	---	---	965
STEP	STEP DEFINE	008	---	---	---	702
STR4	FOUR-DIGIT NUMBER TO ASCII	601	@STR4	---	---	348
STR8	EIGHT-DIGIT NUMBER TO ASCII	602	@STR8	---	---	348
STR16	SIXTEEN-DIGIT NUMBER TO ASCII	603	@STR16	---	---	348
STUP	CHANGE SERIAL PORT SETUP	237	@STUP	---	---	817
SUM	SUM	184	@SUM	---	---	521
SWAP	SWAP BYTES	637	@SWAP	---	---	502
SWRIT	STACK DATA WRITE	640	@SWRIT	---	---	484

T

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
TAN	TANGENT	462	@TAN	---	---	400
TAND	DOUBLE TANGENT	853	@TAND	---	---	448
TANQ	HIGH-SPEED TANGENT	477	@TANQ	---	---	403
TCMP	TABLE COMPARE	085	@TCMP	---	---	185
TIM	HUNDRED-MS TIMER	---	---	---	---	133
TIMH	TEN-MS TIMER	015	---	---	---	137
TIMHX	TEN-MS TIMER	551	---	---	---	137
TIML	LONG TIMER	542	---	---	---	153
TIMLX	LONG TIMER	553	---	---	---	153
TIMU	TENTH-MS TIMER	541	---	---	---	144
TIMUX	TENTH-MS TIMER	556	---	---	---	144
TIMW	HUNDRED-MS TIMER WAIT	813	---	---	---	998
TIMWX	HUNDRED-MS TIMER WAIT	816	---	---	---	998
TIMX	HUNDRED-MS TIMER	550	---	---	---	133
TKOF	TASK OFF	821	@TKOF	---	---	1038
TKON	TASK ON	820	@TKON	---	---	1038
TKY	TEN KEY INPUT	211	@TKY	---	---	729
TMHH	ONE-MS TIMER	540	---	---	---	141
TMHHX	ONE-MS TIMER	552	---	---	---	141
TMHW	TEN-MS TIMER WAIT	815	---	---	---	1003
TMHWX	TEN-MS TIMER WAIT	817	---	---	---	1003
TMUH	HUNDREDTH-MS TIMER	544	---	---	---	147

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
TMUHX	HUNDREDTH-MS TIMER	557	---	---	---	147
TOCV	CONVERT ADDRESS TO CV	285	@TOCV	---	---	977
TPO	TIME-PROPORTIONAL OUTPUT	685	---	---	---	582
TRSET	TIMER RESET	549	@TRSET	---	---	168
TRSM	TRACE MEMORY SAMPLING	045	---	---	---	941
TSR	READ SET TIMER	780	@TSR	---	---	1062
TSW	SET STEP TIMER	781	@TSW	---	---	1062
TTIM	ACCUMULATIVE TIMER	087	---	---	---	150
TTIMX	ACCUMULATIVE TIMER	555	---	---	---	150
TWRIT	WRITE TEXT FILE	704	@TWRIT	---	---	922
TXD	TRANSMIT	236	@TXD	---	---	773
TXDU	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	256	@TXDU	---	---	789

U

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
UP	CONDITION ON	521	---	---	---	66

W

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
WAIT NOT (operand)	ONE CYCLE AND WAIT NOT	805	---	---	---	995
WAIT (input condition)	ONE CYCLE AND WAIT	805	---	---	---	995
WAIT (operand)	ONE CYCLE AND WAIT	805	---	---	---	995
WDT	EXTEND MAXIMUM CYCLE TIME	094	@WDT	---	---	968
WSFT	WORD SHIFT	016	@WSFT	---	---	229

X

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
XCGL	DOUBLE DATA EXCHANGE	562	@XCGL	---	---	214
XCHG	DATA EXCHANGE	073	@XCHG	---	---	214
XCHG$	EXCHANGE STRING	665	@XCHG$	---	---	1027
XFER	BLOCK TRANSFER	070	@XFER	---	---	210
XFERC	BLOCK TRANSFER	565	@XFERC	---	---	1044
XFRB	MULTIPLE BIT TRANSFER	062	@XFRB	---	---	208
XNRL	DOUBLE EXCLUSIVE NOR	613	@XNRL	---	---	360
XNRW	EXCLUSIVE NOR	037	@XNRW	---	---	360
XORL	DOUBLE EXCLUSIVE OR	612	@XORL	---	---	358
XORW	EXCLUSIVE OR	036	@XORW	---	---	358

Z

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ZCP	AREA RANGE COMPARE	088	---	---	---	192
ZCPL	DOUBLE AREA RANGE COMPARE	116	---	---	---	192
ZCPS	SIGNED AREA RANGE COMPARE	117	---	---	---	196
ZCPSL	DOUBLE SIGNED AREA RANGE COMPARE	118	---	---	---	196
ZONE	DEAD ZONE CONTROL	682	@ZONE	---	---	580

Symbols

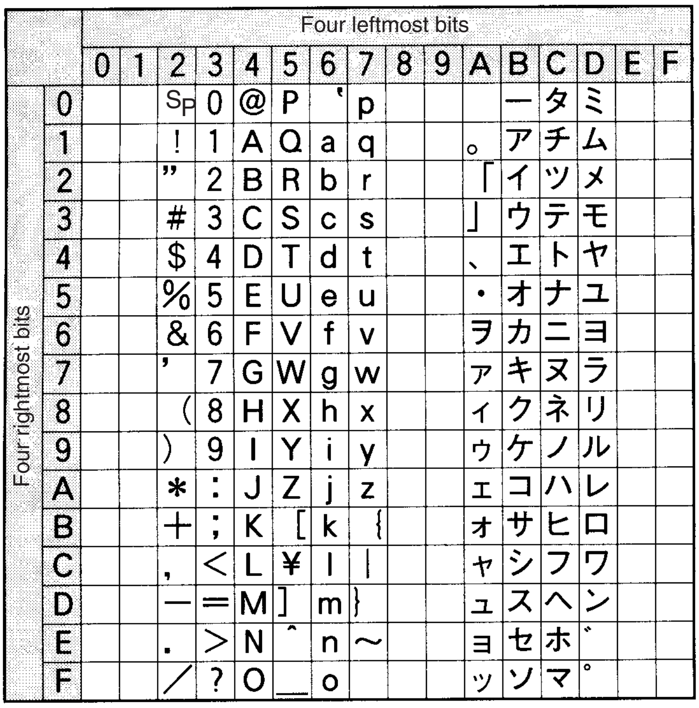
Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
7SEG	7-SEGMENT DISPLAY OUTPUT	214	---	---	---	740
+	SIGNED BINARY ADD WITHOUT CARRY	400	@+	---	---	266
+$	CONCATENATE STRING	656	@+$	---	---	1012
++	INCREMENT BINARY	590	@++	---	---	254
++B	INCREMENT BCD	594	@++B	---	---	260
++BL	DOUBLE INCREMENT BCD	595	@++BL	---	---	260
++L	DOUBLE INCREMENT BINARY	591	@++L	---	---	254
+B	BCD ADD WITHOUT CARRY	404	@+B	---	---	270
+BC	BCD ADD WITH CARRY	406	@+BC	---	---	272
+BCL	DOUBLE BCD ADD WITH CARRY	407	@+BCL	---	---	272
+BL	DOUBLE BCD ADD WITHOUT CARRY	405	@+BL	---	---	270
+C	SIGNED BINARY ADD WITH CARRY	402	@+C	---	---	268
+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	403	@+CL	---	---	268
+D	DOUBLE FLOATING-POINT ADD	845	@+D	---	---	441
+F	FLOATING-POINT ADD	454	@+F	---	---	392
+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	401	@+L	---	---	266
–	SIGNED BINARY SUBTRACT WITHOUT CARRY	410	@–	---	---	274
– –	DECREMENT BINARY	592	@– –	---	---	257
– –B	DECREMENT BCD	596	@– –B	---	---	263
– –BL	DOUBLE DECREMENT BCD	597	@– –BL	---	---	263
– –L	DOUBLE DECREMENT BINARY	593	@– –L	---	---	257
–B	BCD SUBTRACT WITHOUT CARRY	414	@–B	---	---	281
–BC	BCD SUBTRACT WITH CARRY	416	@–BC	---	---	284
–BCL	DOUBLE BCD SUBTRACT WITH CARRY	417	@–BCL	---	---	284

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
−BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	415	@−BL	---	---	281
−C	SIGNED BINARY SUBTRACT WITH CARRY	412	@−C	---	---	278
−CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	413	@−CL	---	---	278
−D	DOUBLE FLOATING-POINT SUBTRACT	846	@−D	---	---	441
−F	FLOATING-POINT SUBTRACT	455	@−F	---	---	392
*	SIGNED BINARY MULTIPLY	420	@*	---	---	286
*B	BCD MULTIPLY	424	@*B	---	---	290
*BL	DOUBLE BCD MULTIPLY	425	@*BL	---	---	290
*D	DOUBLE FLOATING-POINT MULTIPLY	847	@*D	---	---	441
*F	FLOATING-POINT MULTIPLY	456	@*F	---	---	392
*L	DOUBLE SIGNED BINARY MULTIPLY	421	@*L	---	---	286
*U	UNSIGNED BINARY MULTIPLY	422	@*U	---	---	288
*UL	DOUBLE UNSIGNED BINARY MULTIPLY	423	@*UL	---	---	288
−L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	411	@−L	---	---	274
/	SIGNED BINARY DIVIDE	430	@/	---	---	292
/B	BCD DIVIDE	434	@/B	---	---	296
/BL	DOUBLE BCD DIVIDE	435	@/BL	---	---	296
/D	DOUBLE FLOATING-POINT DIVIDE	848	@/D	---	---	441
/F	FLOATING-POINT DIVIDE	457	@/F	---	---	392
/L	DOUBLE SIGNED BINARY DIVIDE	431	@/L	---	---	292
/U	UNSIGNED BINARY DIVIDE	432	@/U	---	---	294
/UL	DOUBLE UNSIGNED BINARY DIVIDE	433	@/UL	---	---	294

Appendix C

ASCII Code Table

ASCII



		Four leftmost bits															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Four rightmost bits	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	ˆ	n	～			ョ	セ	ホ	゛		
	F			／	?	O	_	o				ッ	ソ	マ	゜		

Index

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W474-E1-13

Revision code

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	July 2008	Original production
02	December 2008	Added the CJ-series CJ2 CPU Units (CJ2H-CPU6□) and corrected errors.
03	February 2009	Added information on synchronous unit operation.
04	July 2009	Added the following instructions: ANALOG INPUT DIRECT CONVERSION(AIDC) and ANALOG OUTPUT DIRECT CONVERSION(AODC).
05	September 2009	Added the following instructions: RECEIVE VIA SERIAL COMMUNICATIONS UNIT(DTXDU) and DIRECT TRANSMIT VIA SERIAL COMMUNICATIONS UNIT(DRXDU).
06	October 2009	Added information on the EM Area force-set/reset function to the description of EMBC(281).
07	February 2010	Added the CJ-series CJ2M CPU Units (CJ2M-CPU□□). Added the following instructions. • Data Comparison Instructions: SIGNED AREA RANGE COMPARE (ZCPS) DOUBLE SIGNED AREA RANGE COMPARE (ZCPSL) • I/O Unit Instructions: PCU HIGH-SPEED POSITIONING (NCDMV) PCU POSITIONING TRIGGER (NCDTR) Added information on the CJ2M CPU Units to the following instructions. • Interrupt Control Instructions • Data Comparison Instructions Corrected mistakes.
08	July 2010	• Added INTERRUPT FEEDING (IFEED(892)) instruction. • Added information on using High-Speed Counter and Pulse Output Instructions with the CJ2M CPU Unit. • Corrected mistakes.
09	March 2012	Added information and made minor corrections.
10	March 2015	Corrected mistakes
11	July 2016	Corrected mistakes
12	June 2017	Corrected mistakes
13	December 2018	• Added the following CPU Units. a. CS1D-CPU□□HA b. CS1D-CPU□□SA • Correction of incorrect descriptions